BÙI VIỆT HÀ - BÙI VŨ HUY

Lời giải bài tập PYTHON CƠ BẢN

Thiết lập lớp Circle (vòng tròn) bao gồm thuộc tính R (bán kính) và tọa độ tâm X, Y. Thiết lập các phương thức tính chu vi và diện tích của đối tượng hình tròn.

Lời giải.

```
class Circle:
    def __init__(self,R,x,y):
        self.R = R
        self.x = x
        self.y = y
    def chuvi(self):
        pi = 3.14159
        return 2*self.R*pi
    def dientich(self):
        pi = 3.14159
        return
```

NHÀ XUẤT BẢN ĐẠI HỌC QUỐC GIA HÀ NỘI

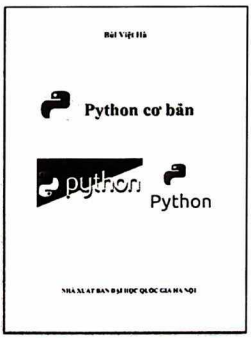
Python là ngôn ngữ lập trình bậc cao đang phát triển rất nhanh và sẽ được đưa vào nhà trường phổ thông và đại học Việt Nam như một chương trình bắt buộc. Trong chương trình môn Tin học của CTGDPT 2018, Python được chính thức khuyến cáo giảng dạy trong phần kiến thức khoa học máy tính, lập trình cho học sinh phổ thông. Cuốn sách này là tuyển tập toàn bộ lời giải các bài tập của cuốn PYTHON CƠ BẢN đã phát hành trước đây.

Sách bao gồm lời giải và chương trình chi tiết của hơn 350 bài tập được phân bổ theo các nội dung sau:

1. Bắt đầu với Python.
2. Làm quen môi trường lập trình Python.
3. Input và chuyển đổi dữ liệu.
4. Hàm số.
5. Đối tượng trong Python.
6. Kiểu dữ liệu List. Mảng một chiều.
7. List của List. Mảng nhiều chiều.
8. Khái niệm Module.
9. Xâu ký tự.
10. Đọc và ghi tệp.
11. Câu lệnh điều kiện.
12. Đệ quy.
13. Kiểu dữ liệu Từ điển.
14. Kiểu dữ liệu Tập hợp.
15. Đồ họa con Rùa.
16. Bắt lỗi và kiểm soát lỗi trong Python.

Bùi Việt Hà – Bùi Vũ Huy

Lời giải bài tập Python cơ bản



Solution Manual

NHÀ XUẤT BẢN ĐẠI HỌC QUỐC GIA HÀ NỘI

Lời nói đầu

Các bạn đang có trên tay **Lời giải bài tập** của cuốn **Python cơ bản** đã phát hành trước đây. Cuốn sách này được viết theo yêu cầu của rất nhiều bạn học sinh và giáo viên muốn có một cuốn sách tập hợp lời giải các bài tập như vậy.

Trong sách này sẽ có đầy đủ lời giải, đáp số, hướng dẫn, chương trình chi tiết cho tất cả 354 bài tập của cuốn sách **Python cơ bản** đã phát hành trước đó. Để phân biệt phần đề bài và lời giải chú ý các điều sau:

- Bài tập của các chương sẽ được đánh số theo chương và số thứ tự bài, ví dụ **5.13** là bài tập số 13 của chương 5. Sau mỗi đề bài là đáp án, đáp số hay lời giải chi tiết của bài tập tương ứng.
- Chương trình chi tiết giải một bài tập cụ thể sẽ luôn ghi rõ tên bài được giải ngay tại dòng đầu tiên của chương trình. Ví dụ lời giải bài tập 5.13 sẽ có dạng:

```
1. # 5.13
2. ... <coding>.....
```

- Trong sách chúng ta có thể phân biệt rõ ràng khuôn dạng thể hiện văn bản khác nhau giữa đề bài và phần lời giải, đáp số, hướng dẫn giải.

Như chúng ta đã biết Chương trình Giáo dục Phổ thông mới 2018 đã có những đổi mới căn bản về học và dạy môn Tin học. Sự thay đổi lớn nhất có lẽ là từ nay trở đi, Tin học trở thành môn học bắt buộc với học sinh phổ thông từ lớp 3 đến lớp 12. Một trong những điểm cơ bản nhất của chương trình môn Tin học mới là nhánh kiến thức Khoa học máy tính (CS – Computer Science) sẽ được đưa vào dạy cho học sinh ngay từ bậc Tiểu học, bên cạnh 2 nhánh kiến thức khác là ICT – Công nghệ Thông tin và DL – học vấn số hóa phổ thông. Việc đưa lập trình vào dạy cho học sinh ngay từ các cấp Tiểu học sẽ làm thay đổi rất nhiều bộ mặt môn Tin học trong nhà trường phổ thông.

Python sẽ đóng vai trò lớn trong môn Tin học. Gần như chắc chắn, Python sẽ được đưa vào chương trình Tin học lớp 10 trong Chương trình môn Tin học mới. Python là ngôn ngữ lập trình bậc cao khá dễ học, rất phù hợp để học và dạy trong nhà trường phổ thông. Python là ngôn ngữ lập trình được sử dụng nhiều nhất trong nghiên cứu và ứng dụng công nghệ mới hiện nay như trí tuệ nhân tạo (AI), Internet vạn vật (IoT), tính toán với số lớn (Big Data). Trong tất cả các trường đại học, các viện nghiên cứu trên thế giới, Python là công cụ không thể thiếu.

Cuốn sách này hướng đến đối tượng là giáo viên dạy Tin học phổ thông, học sinh các cấp THCS và THPT. Hy vọng bộ sách **Python cơ bản** và **Lời giải bài tập Python cơ bản** sẽ có ích cho nhiều học sinh và giáo viên trong việc tiếp cận nhanh đến ngôn ngữ lập trình rất hay và bổ ích này.

Các tác giả

MỤC LỤC

Lời nói đầu 3
LỜI GIẢI CÁC BÀI TẬP 5
Chương 1: Bắt đầu với Python 5
Chương 2. Làm quen với môi trường tương tác Python 8
Chương 3: Input và chuyển đổi dữ liệu 13
Chương 4: Hàm số 21
Chương 5: Đối tượng 29
Chương 6: Kiểu dữ liệu List. Mảng một chiều 34
Chương 7: List của List. Mảng nhiều chiều 45
Chương 8: Khái niệm Module 55
Chương 9: Xâu ký tự 63
Chương 10: Đọc và ghi tệp 77
Chương 11: Câu lệnh điều kiện 85
Chương 12: Đệ quy 98
Chương 13: Kiểu dữ liệu từ điển 112
Chương 14: Kiểu dữ liệu tập hợp 120
Chương 15: Đồ họa con rùa 129
Chương 16: Bắt lỗi và kiểm soát lỗi chương trình 139
Tài liệu tham khảo 143

LỜI GIẢI CÁC BÀI TẬP

Chương 1: Bắt đầu với Python

1.1. Tải phần mềm Wing từ site https://wingware.com/ và cài đặt chương trình để sẵn sàng cho việc lập trình Python trong môi trường này.

1.2. Thiết lập môi trường làm việc Wingware Personal của bạn hỗ trợ hoàn toàn tiếng Việt (cả trong Editor và Python Shell).

1.3. Lệnh `print("Hà Nội","Việt Nam")` sẽ in ra kết quả như thế nào?

A. "Hà Nội" "Việt Nam"

B. Hà Nội Việt Nam

C. Hà NộiViệt Nam

D. Hà Nội

Việt Nam

Đáp án đúng: D.

1.4. Nhóm lệnh `print("Hà Nội"); print("Việt Nam")` sẽ in ra kết quả như thế nào?

A. "Hà Nội" "Việt Nam"

B. Hà Nội Việt Nam

C. Hà NộiViệt Nam

D. Hà Nội

Việt Nam

Đáp án đúng: D.

1.5. Lệnh `print('Xin chào bạn"')` sẽ in ra kết quả như thế nào?

Đáp án: Xin chào bạn"

1.6. Tải phần mềm PyCharm từ link: https://www.jetbrains.com/pycharm/, cài đặt và chạy thử, so sánh với Wingware Personal.

1.7. Tải phần mềm PyScripter từ link sau:

https://sourceforge.net/projects/pyscripter/

Cài đặt, chạy thử và so sánh với 2 phần mềm soạn thảo Python đã biết là Wingware và PyCharm.

1.8. Các lệnh gán sau đây là đúng hay sai trong Python:

A. x = print(x)

B. x = "1234" + 567

C. x = 7, y = 10

D. x = y = z = 10

E. print = print(123)

F. abc = print(0)

Đáp án: A. Đúng; B. Sai; C. Sai; D. Đúng. E. Đúng (không sai); F. Đúng (không sai)

1.9. Biên soạn chương trình đơn giản sau lần lượt trên các trình soạn thảo Wingware, PyCharm và PyScripter.

Hello.py

```
# Chương trình đầu tiên
x = "Hello World!"
print(x)
```

Chạy chương trình trên.

1.10. Theo bạn có thể copy các lệnh từ tệp *.py vào cửa sổ Python Shell được hay không?

Có thể.

1.11. Trong các lệnh sau, lệnh nào là lệnh Python đúng?

A. print("Hello World')

B. Print("Hello World")

C. print(Hello World)

D. print("Hello World")

E. Print('Hello World')

Đáp án: A. Sai; B. Sai; D. Đúng; E. Sai.

1.12. Để viết chú thích cho lệnh trong Python người ta dùng ký tự nào dưới đây:

A. Ký tự //

B. Ký tự /* */

C. Ký tự *

D. Ký tự #

Đáp án đúng: D.

1.13. Có thể lập trình trực tiếp trong môi trường tương tác Python Shell được không?

Trong môi trường Python Shell chỉ có thể lập trình theo từng lệnh.

1.14. Nếu copy nội dung một tệp chương trình *.py vào cửa sổ Python Shell thì có chạy được không?

Không thể nếu chương trình có 2 lệnh trở lên

1.15. Có thể gõ dấu cách phía trước một lệnh Python được không?

Không thể.

1.16. Viết chương trình in ra màn hình bài thơ sau:

Mình về mình có nhớ ta?
Mười lăm năm ấy thiết tha mặn nồng.
Mình về mình có nhớ không?
Nhìn cây nhớ núi, nhìn sông nhớ nguồn.

```
# 1.16
print("""Mình về mình có nhớ ta?
Mười lăm năm ấy thiết tha mặn nồng.
Mình về mình có nhớ không?
Nhìn cây nhớ núi, nhìn sông nhớ nguồn.""")
```

1.17. Viết chương trình trên Python tính tổng 1 + 2 + 3 + ... + 10.

```
# 1.17
print(1+2+3+4+5+6+7+8+9+10)
```

1.18. Viết chương trình trên Python tính biểu thức $1^1 + 2^2 + 3^3 + 4^4 + 5^5$.

```
# 1.18
x = 1 + 2*2 + 3*3*3 + 4*4*4*4 + 5*5*5*5*5
print(x)
```

1.19. Lệnh print("Hà Nội", "thủ đô","Việt Nam", sep = "\n", end = " ") sẽ in ra kết quả như thế nào?

```
# 1.19
>>> print("Hà Nội", "thủ đô","Việt Nam", sep = "\n", end = " ")
Hà Nội
thủ đô
Việt Nam
```

1.20. Các lệnh đã nhập trong Python Shell có thể sao chép sang cửa sổ soạn thảo lập trình được không?

Được, có thể sao chép lệnh từ Python Shell sang cửa sổ lập trình.

Chương 2. Làm quen với môi trường tương tác Python

2.1. Em hãy giải thích vì sao một biến nhớ trong Python tại các thời điểm khác nhau có thể gán các giá trị dữ liệu có kiểu khác nhau.

Giải thích: Python kiểm soát các tên biến nhớ và đối tượng dữ liệu độc lập. Khi thực hiện lệnh gán, thực chất Python sẽ "gán" đối tượng dữ liệu lên biến nhớ. Ví dụ khi thực hiện lệnh:

```
x = 2
```

Python sẽ thực hiện việc gắn tên x với đối tượng số nguyên 2. Nếu tại thời điểm khác, chúng ta thực hiện lệnh:

```
x = "Hà Nội"
```

Python sẽ gán tên x với đối tượng xâu ký tự "Hà Nội".

2.2. Các lệnh sau đây có hợp lệ không?

A. x = y = z = 1

B. x = 1, y = 2, z = 3

C. x, y = 1

D. x, y, z = 1, 2, 3

Đáp án: A. Hợp lệ; B: Không hợp lệ; C: Không hợp lệ; D: Hợp lệ.

2.3. Sau các lệnh:

```
>>> x = y = 1
>>> x = x + 1
```

thì giá trị y bằng bao nhiêu?

Đáp án: x = 2; y = 1

2.4. Sau các lệnh:

```
>>> x = 10
>>> y = x
>>> x = 20
```

thì giá trị y bằng bao nhiêu?

Trả lời: x = 20; y = 10.

2.5. Sau các lệnh:

```
>>> x = y
>>> y = x
```

Thì giá trị các biến nhớ x, y thay đổi như thế nào so với trước khi thực hiện các lệnh này?

Trả lời: x, y đổi giá trị cho nhau sau các lệnh trên.

2.6. Có thể đặt tên biến nhớ trùng với từ khóa của Python được hay không?

Không thể.

2.7. Lệnh:

```
>>> print(13 + 25)
```

sẽ in ra kết quả là gì?

a) 13 + 25

b) '13 + 25'

c) 38

d) '38'

Đáp án đúng: c).

2.8. Đặt tên biến nhớ như sau có được không?

a) a-b-c

b) a_b_c

c) __abc__

d) -a_b_c-

Đáp án: a) sai; b) đúng; c) đúng; d) sai.

2.9. Các tên định danh sau có hợp lý không trong Python?

a) 1_abc

b) A.B.C

c) ____1_

d) 1.abc

e) a&b&c

Đáp án: a) không; b) không; c) có; d) không; e) không.

2.10. Đặt tên các biến nhớ như sau có hợp lệ không?

a) None

b) none

c) Việt Nam

d) x y z

e) break

f) happy-day

Đáp án: a) không; b) có; c) không; d) không; e) không; f) không.

2.11. Hai biểu thức x ** 2 và x * 2 có đồng nhất giá trị không với các giá trị khác nhau của x là nguyên hay thập phân?

Trả lời: Không.

2.12. Hai biểu thức x ** 2.0 và x * x có đồng nhất giá trị không với các giá trị khác nhau của x là nguyên hay thập phân?

Trả lời: Không.

2.13. Kết quả của lệnh, biểu thức print(1 – 4 % 6) như thế nào?

a) 1 – 4 % 6

b) -3

c) 3

Đáp án đúng: b).

2.14. Biểu thức print(- 4 % 6) có kết quả như thế nào?

a) – 4 % 6

b) – 4

c) 2

Đáp án đúng: c).

2.15. Hai biểu thức (2 ** 3) ** 4 và 2 ** 3 ** 4 có giá trị giống nhau hay khác nhau?

Trả lời: khác nhau.

*(2**3)**4 = 4096*

*2**3**4 = 2417851639229258349412352*

2.16. Hãy kiểm tra xem công thức:

a = b * (a // b) + (a % b).

có còn đúng không nếu a, b không là số nguyên.

Trả lời: Luôn đúng.

2.17. Nếu a, b đồng thời là số nguyên thì giá trị các biểu thức a / b và a // b có đồng nhất với nhau không?

Trả lời: Không.

2.18. Giả sử ss = giá trị tính bằng số giây cho trước. Khi đó muốn đổi giá trị này ra phút và giờ thì dùng các công thức tính nào? Chú ý chỉ cần tính tròn phút và tròn giờ.

Công thức tính, đổi giây thành phút (làm tròn):

ss // 60

Công thức tính, đổi giây thành giờ (làm tròn):

ss // 3600

2.19. Giả sử có đoạn chương trình sau:

```
>>> a = 5
>>> b = 7
>>> a, b = b, a
```

Sau các lệnh trên giá trị các biến nhớ a, b bằng bao nhiêu?

a) a = 5, b = 5

b) a = 5, b = 7

c) a = 7, b = 7

d) a = 7, b = 5

Đáp án đúng: d).

2.20. Công thức tính theo radian (rad) của một góc tính bằng độ (angle) như sau:

Rad = angle × 3.14 / 180.

Viết lệnh bằng Python mô tả công thức trên.

```
# 2.20
Rad = angle * (3.14/180)
```

2.21. Sau các lệnh sau thì x, y nhận các giá trị nào?

```
>>> x = 2
>>> y = 5
>>> x += 1
>>> x, y = y, x
>>> y += 1
```

Đáp số: x = 5; y = 4.

2.22. Trên thế giới có 2 cách thể hiện nhiệt độ: Nhiệt độ C theo Celsius và nhiệt độ K theo Kenvin. Công thức tính nhiệt độ T theo Kenvin tính theo nhiệt độ t theo Celsius như sau:

T = t + 273.

Viết công thức trong Python tính nhiệt độ Kenvin theo Celsius và ngược lại.

Lời giải.

Công thức tính nhiệt độ Kenvin (K) theo Celsius (C):

K = C + 273

Công thức tính nhiệt độ Celsius (C) theo Kenvin (K):

C = K – 273.

2.23. Có thể tính biểu thức 1 + 2 + 3 + ... + 10 chỉ bằng 4 lệnh trong Python hay không, trong đó tất cả các biểu thức tính toán của các lệnh này có không quá 2 phép cộng?

Lời giải.

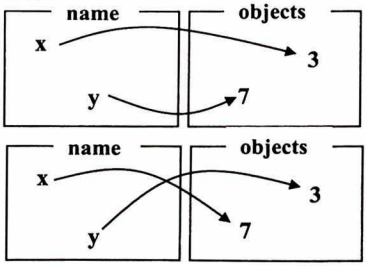
Thực hiện 4 lệnh sau đây:

```
x = 1 + 2 + 3
x = x + 5 + 6
x = x + 7 +8
x = x + 9 + 10
```

2.24. Dùng mô hình biểu diễn **Namespace** để giải thích kết quả lệnh:

x, y = y, x

Giải thích: Giả sử trước khi thực hiện lệnh trên, các biến nhớ x, y nhận các giá trị lần lượt là 3, 7



*Trước khi thực hiện lệnh, tên **x** được trỏ vào đối tượng 3, tên **y** được trỏ vào đối tượng 7.*

*Sau khi thực hiện lệnh, tên **x** được trỏ vào đối tượng 7 (là đối tượng trước đây của **y**), tên **y** được trỏ vào đối tượng 3 (là đối tượng trước đây của **x**).*

2.25. Cho trước các giá trị s tính bằng giây, m tính bằng phút, h tính bằng giờ, d tính bằng ngày. Viết công thức đổi giá trị "d ngày, h giờ, m phút, s giây" ra:

a) giây.

b) phút.

c) giờ.

Lời giải.

a) Công thức đổi ra giây:

*d*24*3600 + h*3600 + m*60 + s*

b) Công thức đổi ra phút:

*d*24*60 + h*60 + m + s/60*

c) Công thức đổi ra giờ:

*d*24 + h + m/60 + s/3600*

```
# 2.25
s = 10
m = 2
h = 3
d = 1

# Công thức đổi ra giây:
ss = d*24*3600 + h*3600 + m*60 + s
print("Số giây:",ss)

# Công thức đổi ra phút:
mm = d*24*60 + h*60 + m + s/60
print("Số phút:",mm)

# Công thức đổi ra giờ:
hh = d*24 + h + m/60 + s/3600
print("Số giờ:",hh)
```

Chương 3: Input và chuyển đổi dữ liệu

3.1. Các biểu thức sau sẽ có giá trị đúng hay sai?

a) 1 < 2 == 3 < 4

b) 4 < 4 - 1

c) False == False

d) True == False

Đáp án: a) đúng; b) sai; c) đúng; d) sai.

3.2. Giả sử thực hiện dòng lệnh sau trên Python:

```
x = input("Enter x: ")
```

Khi thực hiện người dùng sẽ nhập từ dòng yêu cầu nhập như sau: 3 + 4 * 5

Hỏi giá trị x khi đó là gì?

a) 3 + 4 * 5

b) 23

c) "3 + 4 * 5"

Đáp án đúng: c).

3.3. Giả sử em thực hiện các lệnh sau từ cửa sổ tương tác của Python, sau đó nhập dữ liệu tại chỗ:

```
>>> x = int(input("Nhập x: "))
Nhập x: 3 - 4/7 + 2**3
```

Kết quả sau các lệnh trên x nhận giá trị nào?

a) 10

b) 11

c) Python thông báo lỗi.

Đáp án đúng: c).

3.4. Giả sử em thực hiện các lệnh sau từ cửa sổ tương tác của Python, sau đó nhập dữ liệu tại chỗ:

```
>>> x = int(input("Nhập x: "))
Nhập x: 5.0
```

Kết quả sau các lệnh trên x nhận giá trị nào?

a) 5

b) 5.0

c) '5'

d) Python thông báo lỗi.

Đáp án đúng: a).

3.5. Giả sử em thực hiện các lệnh sau từ cửa sổ tương tác của Python, sau đó nhập dữ liệu tại chỗ:

```
>>> x = float(input("Nhập x: "))
Nhập x: 4
```

Kết quả sau các lệnh trên x nhận giá trị nào?

a) 4

b) 4.0

c) '4.0'

d) Python thông báo lỗi.

Đáp án đúng: b).

3.6. Mệnh đề sau đúng hay sai:

Hàm eval() chấp nhận trong tham số của mình tất cả các biểu thức sau:

"2 + 2**5", "3+ 4/3, 5 ** 2", "3.456 + 2.890".

Mệnh đề trên đúng.

3.7. Viết chương trình nhập 2 số tự nhiên m, n trên một dòng cách nhau bởi dấu phẩy, sau đó hiển thị tổng và tích 2 số này.

```
# 3.7
a,b = eval(input("Nhập 2 số cách nhau bởi dấu phảy: "))
print("Tổng 2 số:",a+b)
print("Tích 2 số:",a*b)
```

3.8. Viết chương trình nhập 3 số dương a, b, c trên một dòng cách nhau bởi dấu phẩy, kiểm tra xem 3 số này có tạo thành độ dài của một tam giác được hay không, thể hiện trên màn hình dòng chữ "3 số a, b, c tạo thành một tam giác" hoặc "3 số a, b, c không tạo thành một tam giác".

```
# 3.8
a,b,c = eval(input("Nhập 3 số cách nhau bởi dấu phảy: "))
if 0 < a < b + c and 0 < b < a+c and 0 < c < a+b:
    print("3 số a, b, c tạo thành một tam giác.")
else:
    print("3 số a,b,c không tạo thành tam giác.")
```

3.9. Năm nhuận là năm mà giá trị của nó thỏa mãn điều kiện: Chia hết cho 400 hoặc chia hết cho 4 nhưng không chia hết cho 100. Viết biểu thức logic để kiểm tra giá trị số **year** có là năm nhuận hay không.

Biểu thức logic sau kiểm tra năm n có là nhuận hay không.

```
n%400 == 0 or (n%4 == 0 and n%100 != 0)
```

310. Viết chương trình nhập 3 số tự nhiên từ bàn phím là **day**, **month** và **year** có ý nghĩa là ngày, tháng, năm tương ứng. Kiểm tra xem bộ dữ liệu đã nhập có hợp lý hay không.

```
1.  # 3.10
2.  # Nhập và kiểm tra bộ 3 số day, month, year có hợp lệ không?
3.  day,month,year = eval(input("Nhập 3 số ngày, tháng, năm: "))
4.  nhuan = year%400 == 0 or (year%4 == 0 and year%100 != 0)
5.  if not (1 <= day <= 31 and 1 <= month <= 12 and year > 0):
6.      print("Số liệu đã nhập không hợp lệ.")
7.  else:
8.      if (month == 4 or month == 6 or month == 9 or month == 11)
    and day == 31:
```

```
9.          print("Số liệu đã nhập không hợp lệ.")
10.     elif nhuan and month == 2 and day > 29:
11.         print("Số liệu đã nhập không hợp lệ.")
12.     elif (not nhuan) and month == 2 and day > 28:
13.         print("Số liệu đã nhập không hợp lệ.")
14.     else:
15.         print("Số liệu đã nhập hoàn toàn hợp lệ.")
```

311. Viết chương trình nhập 3 số tự nhiên từ bàn phím là **day**, **month** và **year** có ý nghĩa là ngày, tháng, năm tương ứng. Viết chương trình tính tổng số ngày tính từ đầu năm đó (ngày 1/1) cho đến ngày hiện thời. Chú ý đến yếu tố năm nhuận[1].

```
1.  # 3.11
2.  day,month,year = eval(input("Nhập 3 số ngày, tháng, năm: "))
3.  b = False
4.  nhuan = year%400 == 0 or (year%4 == 0 and year%100 != 0)
5.  T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,T11,T12                    =
    31,28,31,30,31,30,31,31,30,31,30,31
6.  if nhuan:
7.      T2 = 29
8.  if not (1 <= day <= 31 and 1 <= month <= 12 and year > 0):
9.      print("Số liệu đã nhập không hợp lệ.")
10. else:
11.     if (month == 4 or month == 6 or month == 9 or month == 11)
    and day == 31:
12.         print("Số liệu đã nhập không hợp lệ.")
13.     elif nhuan and month == 2 and day > 29:
14.         print("Số liệu đã nhập không hợp lệ.")
15.     elif (not nhuan) and month == 2 and day > 28:
16.         print("Số liệu đã nhập không hợp lệ.")
17.     else:
18.        b = True
19. if b:
20.     if month == 1:
21.         num = day
22.     if month == 2:
23.         num = T1 + day
24.     if month == 3:
25.         num = T1 + T2 + day
26.     if month == 4:
27.         num = T1 + T2 + T3 + day
28.     if month == 5:
29.         num = T1 + T2 + T3 + T4 + day
30.     if month == 6:
31.         num = T1 + T2 + T3 + T4 + T5 + day
32.     if month == 7:
33.         num = T1 + T2 + T3 + T4 + T5 + T6 + day
34.     if month == 8:
35.         num = T1 + T2 + T3 + T4 + T5 + T6 + T7 + day
36.     if month == 9:
37.         num = T1 + T2 + T3 + T4 + T5 + T6 + T7 + T8 + day
38.     if month == 10:
```

[1] Lời giải bài tập này chỉ được phép sử dụng các kiến thức của các bài học 1, 2, 3. Nếu sử dụng kiến thức của các bài học sau, ví dụ list thì lời giải có thể viết gọn hơn rất nhiều.

```
39.          num = T1 + T2 + T3 + T4 + T5 + T6 + T7 + T8 + T9 + da
40.      if month == 11:
41.          num = T1 + T2 + T3 + T4 + T5 + T6 + T7 + T8 + T9 + T1
    + day
42.      if month == 12:
43.          num = T1 + T2 + T3 + T4 + T5 + T6 + T7 + T8 + T9 + T1
    + T11 + day
44.      print("Số ngày tính từ đầu năm",year,"đế
    ngày",day,"/",month,"/",year,"là:",num)
```

3.12. Viết chương trình nhập số tự nhiên n từ bàn phím, sau đó kiểm tra xem số n có phả là số nguyên tố hay không, thông báo trên màn hình.

```
1.  # 3.12
2.  n = int(input("Nhập số n: "))
3.  s = 0
4.  for i in range(2, n):
5.      if n % i == 0:
6.          s = s + 1
7.  if s == 0:
8.      print(n, "là số nguyên tố")
9.  else:
10.     print(n,"không là số nguyên tố")
```

3.13. Viết chương trình nhập từ bàn phím số tự nhiên ss tính theo giây, sau đó đổi số giâ này ra thành giờ, phút, giây và thông báo trên màn hình. Ví dụ nếu nhập vào ss = 374 thì kết quả hiển thị như sau:

3740 giây = 1 giờ 2 phút 20 giây.

```
# Bài 3.13
ss = int(input("Nhập số giây gốc: "))
h,d = divmod(ss,3600)
m,s = divmod(d,60)
print(ss,"giây = ",h,"giờ",m,"phút",s,"giây")
```

3.14. Viết chương trình nhập số tự nhiên n và in ra trên màn hình tất cả các ước số thự sự của n.

```
# 3.14
n = int(input("Nhập số tự nhiên n: "))
for k in range(1,n):
    if n%k == 0:
        print(k, end = " ")
```

3.15. Viết chương trình nhập số tự nhiên n và đếm số các ước số thực sự của n. In kết qu ra màn hình.

```
# 3.15
n = int(input("Nhập số tự nhiên n: "))
count = 0
for k in range(1,n):
    if n%k == 0:
        count = count + 1
print(count)
```

3.16. Viết các chương trình nhập số tự nhiên n và tính các giá trị sau:

a) Tính tổng 1 + 2 + ... + 2n.

b) Tính tổng các số tự nhiên < n và là số lẻ.

c) Tính tổng các số tự nhiên < n và là số chẵn.

```
# 3.16
n = int(input("Nhập số tự nhiên n: "))
S1 = 0
S2 = 0
S3 = 0
for k in range(1,2*n + 1):
    S1 = S1 + k
for k in range(1,n):
    if k%2 == 1:
        S2 = S2 + 1
    else:
        S3 = S3 + 1
print("Câu a:", S1)
print("Câu b:", S2)
print("Câu c:", S3)
```

3.17. Viết các chương trình nhập một số tự nhiên n và tính các giá trị sau:

a) Tính số các ước số nguyên tố khác nhau của n (tính cả n).

b) Tính tổng các ước số thực sự của n (không tính n).

c) Tính tổng $1^2 + 2^2 + 3^2 + ... + n^2$.

Câu a).

```
# 3.17a
n = int(input("nhap 1 so n : "))
count = 0
i = 2
while n > 1:
    if n % i != 0:
        i = i + 1
    else:
        print(i,end = " ")
        count = count + 1
        while n%i == 0:
            n = n//i
print()
print("Số các ước nguyên tố là:",count)
```

Câu b).

```
# 3.17b
n = int(input("Nhập số n: "))
s = 0
for i in range(1,n):
    if n % i == 0:
        s = s +i
print(s)
```

Câu c).

```
# 3.17c
n = int(input("Nhập số n: "))
s = 0
```

```
4. for i in range(1,n+1):
5.     s = s + i*i
6. print(s)
```

3.18. Cho trước biến nhớ p. Làm thế nào để biết được biến nhớ p thuộc kiểu int, float hay str?

Lời giải.

*- Biến nhớ p thuộc kiểu **int** nếu thỏa mãn một trong các điều kiện sau:*

```
str(type(p)) == "<class 'int'>"
type(p) == type(0)
```

*- Biến nhớ p thuộc kiểu **float** nếu thỏa mãn một trong các điều kiện sau:*

```
str(type(p)) == "<class 'float'>"
type(p) == type(0.0)
```

*- Biến nhớ p thuộc kiểu **string** nếu thỏa mãn một trong các điều kiện sau:*

```
str(type(p)) == "<class 'str'>"
type(p) == type("")
```

3.19. Viết chương trình nhập số tự nhiên n từ bàn phím. Tính và kiểm tra tính đúng đắn của biểu thức sau:

$$1^2 + 2^2 + 3^2 + ... + n^2 = \frac{n(n+1)(2n+1)}{6}$$

```
1. # 3.19
2. n = int(input("Nhập số tự nhiên n: "))
3. s1 = n*(n+1)*(2*n+1)//6
4. s2 = 0
5. for k in range(1,n+1):
6.     s2 = s2 + k*k
7. print("1^2 + 2^2 + ... + n^2 =",s2)
8. print("n(n+1)(2n+1)/6 =",s1)
```

3.20. Viết chương trình nhập số tự nhiên n từ bàn phím. Tính và kiểm tra tính đúng đắn của biểu thức sau:

$$1^3 + 2^3 + 3^3 + ... + n^3 = (1 + 2 + 3 + ... + n)^2$$

```
1. # 3.20
2. n = int(input("Nhập số tự nhiên n: "))
3. s1 = 0
4. s2 = 0
5. for k in range(1,n+1):
6.     s1 = s1 + k*k*k
7. for k in range(1,n+1):
8.     s2 = s2 + k
9. s2 = s2*s2
10. print("1^3 + 2^3 + ... + n^3 =",s1)
11. print("(1 + 2 + ... + n)^2 =",s2)
```

3.21. Viết chương trình đếm số các số nguyên tố < 10000.

```
1. # 3.21
2. # in và đếm số các số nguyên tố < n
3. n = int(input("Nhập số tự nhiên n: "))
4. count = 0
5. for num in range(n):
```

```
6.      if num > 1:
7.          for i in range(2, num):
8.              if num % i == 0:
9.                  break
10.         else:
11.             #print(num, end = " ")
12.             count = count + 1
13. print("Có",count,"số nguyên tố <",n)
```

3.22. Viết chương trình nhập số tự nhiên n từ bàn phím và in ra số nguyên tố nhỏ nhất lớn hơn hoặc bằng n.

```
1.  # 3.22
2.  n = int(input("Nhập số n : "))
3.  num = n
4.  found = False
5.  while not found:
6.      for k in range(2,num):
7.          if num%k == 0:
8.              break
9.      if num%k == 0:
10.         num = num + 1
11.     else:
12.         found = True
13. print(num)
```

3.23. Số tự nhiên được gọi là hoàn hảo nếu số đó bằng tổng các ước số thực sự của mình. Ví dụ số 6 là hoàn hảo vì 6 = 1 + 2 + 3. Viết chương trình nhập số tự nhiên n từ bàn phím và kiểm tra xem n có phải là số hoàn hảo hay không. Thông báo trên màn hình.

```
1.  # 3.23
2.  n = int(input("Nhập số n : "))
3.  s = 0
4.  for i in range(1, n):
5.      if n % i == 0:
6.          s = s + i
7.  if s == n:
8.      print(n,"là số hoàn hảo")
9.  else:
10.     print(n, "không là số hoàn hảo")
```

3.24. Viết chương trình tìm tất cả các số tự nhiên hoàn hảo < 10000.

Lời giải tổng quát: Nhập số tự nhiên n từ bàn phím và in ra tất cả các số hoàn hảo < n.

```
1.  # 3.24
2.  n = int(input("Nhập số n : "))
3.  for i in range(1, n):
4.      s = 0
5.      for j in range(1, i):
6.          if i % j == 0:
7.              s = s + j
8.      if s == i:
9.          print(s)
```

3.25. Viết chương trình in ra tất cả các cặp số nguyên tố cách nhau 2 đơn vị và < 10000. Ví dụ các cặp số nguyên tố đầu tiên cách nhau 2 đơn vị là (3, 5), (5, 7), (11, 13).

Lời giải sau sẽ yêu cầu nhập số tự nhiên n từ bàn phím và tìm tất cả các cặp số nguyên tố sinh đôi (cặp số nguyên tố là các số tự nhiên cách nhau 2 đơn vị) nhỏ hơn n.

```
1.  # 3.25
2.  n = int(input("Nhập số n: "))
3.  for k in range(3,n):
4.      m1,m2 = k,k+2
5.      b1,b2 = False, False
6.      i1 = 2
7.      while i1 < m1 and m1%i1 != 0:
8.          i1 += 1
9.      b1 = i1 == m1
10.     i2 = 2
11.     while i2 < m2 and m2%i2 != 0:
12.         i2 += 1
13.     b2 = i2 == m2
14.     if b1 and b2:
15.         print((m1,m2))
```

Chương 4: Hàm số

4.1. Sau khi thực hiện đoạn chương trình sau thì giá trị **kq** là bao nhiêu?

```
1. kq = 1
2. def calc(x):
3.     kq = 4*x-1
4.     return kq
5. calc(7)
```

a) 27

b) 1

c) 7

d) 0

Đáp án đúng: b.

4.2. Đoạn chương trình sau sẽ in ra kết quả bao nhiêu:

```
1. def f(x):
2.     return x ** 2, x**x**x
3. x, y = f(2)
4. print(x+y)
```

a) 10

b) 20

c) 16

d) Chương trình báo lỗi.

Đáp án đúng: b.

4.3. Viết các hàm **lambda** mô tả các biểu thức sau:

a) x + y + z

b) x ** (y + z)

c) (x + y) ** 2

Lời giải.

```
f_a = lambda x,y,z: x+y+z
f_b = lambda x,y,z: x**(y+z)
f_c = lambda x,y: (x+y)**2
```

4.4. Viết hàm số f(n) với tham số n là số tự nhiên theo yêu cầu sau.

a) Hàm trả lại số các ước số thực sự của n.

b) Hàm trả lại 1 nếu n là nguyên tố, ngược lại hàm trả về 0.

c) Hàm đếm số các ước số là lẻ của n.

d) Hàm đếm số các ước số nguyên tố của n.

e) Hàm tính tổng tất cả các ước số thực sự của n.

Câu a).

```
1. # Bài 4.4a. Hàm trả lại số các ước số thực sự của n.
```

```
def USTS(n):
    count = 0
    for k in range(1,n):
        if n%k == 0:
            count = count + 1
    return count
```

Câu b).

```
# Bài 4.4b.
# Hàm trả lại 1 nếu n là nguyên tố, ngược lại hàm trả về 0.
def prime(n):
    count = 0
    for k in range(1,n):
        if n%k == 0:
            count = count + 1
    if count == 1:
        return 1
    else:
        return 0
```

Câu c).

```
# Bài 4.4c
# Hàm đếm số các ước số là lẻ của n.
def USLE(n):
    count = 0
    for k in range(1,n+1):
        if n%k == 0 and k%2 == 1:
            count = count + 1
    return count
```

Câu d).

```
# Bài 4.4d.
# Hàm đếm số các ước số nguyên tố của n.
def prime(n):
    count = 0
    for k in range(1,n):
        if n%k == 0:
            count = count + 1
    if count == 1:
        return 1
    else:
        return 0
def primecount(n):
    count = 0
    for m in range(1,n+1):
        if n%m == 0 and prime(m) == 1:
            count = count + 1
    return count
```

Câu e).

```
# Bài 4.4e
# Hàm tính tổng tất cả các ước số thực sự của n.
def Sum_USTS(n):
    count = 0
    for k in range(1,n):
```

```
6.          if n%k == 0 and k%2 == 1:
7.              count = count + 1
8.      return count
```

4.5. Viết hàm số f(m, n) với m, n là số tự nhiên theo yêu cầu sau.

a) Hàm đếm số các ước chung của các số m, n.

b) Hàm đếm số các ước số của m nhưng không là ước số của n.

c) Hàm tính tổng các ước số chung của m và n.

Câu a).

```
def f(m,n):
    count = 0
    for k in range(1,min(m,n)):
        if m%k == 0 and n%k == 0:
            count = count + 1
    return count
```

Câu b).

```
def f(m,n):
    count = 0
    for k in range(1,min(m,n)):
        if m%k == 0 and n%k != 0:
            count = count + 1
    return count
```

Câu c).

```
def f(m,n):
    S = 0
    for k in range(1,min(m,n)):
        if m%k == 0 and n%k == 0:
            S = S + k
    return S
```

4.6. Viết hàm số Radian(d), tham số đầu vào là số thực là số đo góc theo độ bình thường. Hàm sẽ trả lại số đo của d theo radian. Công thức chuyển đổi là $R = d \times \pi / 180$.

Lời giải.

```
def Radian(d):
    pi = 3.14159
    return d*pi/180
```

4.7. Viết hàm số Degree(R), tham số đầu vào là số đo góc theo đơn vị radian. Hàm sẽ trả lại số đo góc theo độ bình thường. Công thức chuyển đổi là: $d = R \times 180/\pi$.

Lời giải.

```
def Degree(R):
    pi = 3.14159
    return R*180/pi
```

4.8. Sau đoạn chương trình sau, biến nhớ R nhận giá trị nào?

```
>>> def f(x,y,R):
      R = x**2 + y**2
      return
>>> R = 1
>>> f(3,4,R)
```

a) 25

b) 1

c) 0

d) Ngẫu nhiên.

Đáp án đúng: b).

4.8. Một hàm (thủ tục) khi khai báo có 3 tham số, ví dụ, f(x, y, z). Nhưng khi gọi hàn lời gọi chỉ có 2 tham số, ví dụ f(x,y). Điều đó có thể thực hiện được hay không tron Python?

Được, nếu khai báo tham số z trong hàm f() là tham số lựa chọn. Ví dụ:

```
def f(x, y, z = 0)
```

4.9. Hàm f(x,y) được xác định như sau:

```
>>> def f(x,y):
      return x**3 + y**2
```

Khi đó lời gọi hàm f(y=1,x=3) sẽ trả lại giá trị nào?

a) 10

b) 28

c) Chương trình báo lỗi.

Đáp án đúng: b).

4.10. Viết thủ tục **convert()** thực hiện các công việc sau:

- Yêu cầu nhập từ bàn phím số giờ, phút, giây (tính theo số nguyên).
- Sau đó đổi tổng số thời gian này ra ngày (tính theo float).
- Hiển thị kết quả ra màn hình.

Ví dụ thủ tục này chạy như sau.

```
Nhập số giờ: 100
Nhập số phút: 43
Nhập số giây: 56
4.197175925925926
```

Lời giải.

Ta có:

1 phút = 60 giây.

1 giờ = 60 phút = 3600 giây.

1 ngày = 24 giờ = 1440 phút = 86400 giây.

Chương trình như sau:

```
1. # 4.10
2. h = int(input("Nhập số giờ: "))
3. m = int(input("Nhập số phút: "))
4. s = int(input("Nhập số giây: "))
5. ngay = h/24 + m/1440 + s/86400
6. print(ngay)
```

4.11. Viết hàm số **your_weight(w, m)** dùng để tính trọng lượng (hay cân nặng) thực tế của bạn trên một hành tinh nào đó. Hàm này có 2 tham số:

- w: cân nặng của bạn trên Trái đất, tính theo kg.

- m: Trọng lực (Gravity) của hành tinh mà bạn cần tính, giá trị này tính theo đơn vị mét/giây^2 (m/s^2).

Chú ý rằng gravity của Trái đất là 9.8m/s^2.

Công thức tính trọng lượng của bạn trên hành tinh có gravity = m được tính như sau:

Cân nặng của bạn sẽ là w*m/9.8.

Ví dụ hàm số này chạy như sau:

```
>>> your_weight(70.3,9.8)   # trọng lượng trên Trái đất
70.3
>>> your_weight(70.3,23.1)  # trọng lượng trên sao Mộc.
165.70714285714286
```

Lời giải.

```
def your_weight(w,m):
    return w*m/9.8
```

4.12. Chúng ta đã biết đơn vị đo nhiệt độ hiện nay trên thế giới có 3 loại:

- Nhiệt độ tính theo Celsius (nhiệt độ C).

- Nhiệt độ tính theo Kelvin (nhiệt độ K).

- Nhiệt độ tính theo Fahrenheit (nhiệt độ F).

Các công thức sau cho biết quan hệ toán học giữa các nhiệt độ trên.

$T_c = \frac{5}{9}(T_f - 32)$

$T_k = T_c + 273.15$

Viết thủ tục **convertCK()** thực hiện các công việc sau:

- Yêu cầu nhập từ màn hình giá trị nhiệt độ T_c tính theo C.

- Tính toán và đưa ra màn hình nhiệt độ T_k tính theo K.

Lời giải.

Từ các công thức trên suy ra các công thức sau tính quan hệ qua lại giữa 3 đại lượng Tc, Tk, Tf như sau:

1) T_k, T_f tính qua T_c:

$T_k = T_c + 273.15$

$T_f = \frac{9}{5}T_c + 32$

2) Tc, Tk tính qua T_f:

$T_c = \frac{5}{9}(T_f - 32)$

$T_k = T_c + 273.15 = \frac{5}{9}(T_f - 32) + 273.15 = \frac{5}{9}T_f + 255.37$

3) T_c, T_f tính qua T_k:

$T_c = T_k - 273.15$

$T_f = \frac{9}{5}T_c + 32 = = \frac{9}{5}(T_k - 273.15) + 32 = \frac{9}{5}T_k - 459.67$

Hàm chuyển đổi convertCK() viết như sau:

```
def convertCK(t):
    return t + 273.15
```

4.13. Tương tự viết các thủ tục chuyển đổi nhiệt độ **convertKC()**, **convertFK** **convertKF()**, **convertFC()** và **convertCF()**.

Các hàm chuyển đổi nhiệt độ được viết như sau, suy từ các công thức được tính tro bài tập trên.

```
def convertKC(t):
    return t - 273.15
def convertFK(t):
    return t*5/9 + 255.37
def convertKF(t):
    return t*9/5 - 459.67
def convertFC(t):
    return (t - 32)*5/9
def convertCF(t):
    return t*9/5 + 32
```

4.14. Đoạn chương trình sau có lời gọi hàm f(x) ở cuối.

```
>>> def f(x):
      return x+y+z
>>> f(1)
```

Kết quả thực hiện trả lại giá trị nào?

a) 1

b) 0

c) Chương trình báo lỗi.

Đáp án đúng: c).

4.15. Viết các thủ tục thực hiện các công việc sau:

a) Tham số là số tự nhiên n, in ra màn hình kết quả tổng các số 1 + 2 + ... + n.

b) Tham số là 2 số tự nhiên m, n, in ra thông báo: "Hai số này là nguyên tố cùng nha hoặc "Hai số này không là nguyên tố cùng nhau".

c) Tham số là số tự nhiên s chỉ số giây, thông báo số giờ, phút, giây mà giá trị s ı chuyển sang.

d) Tham số là hai số thực a, b, trả lại giá trị là trị tuyệt đối | a - b |.

Câu a).

```
def Summary(n):
    sum = 0
    for i in range(1,n+1):
        sum += i
    return sum
```

Câu b).

```
def HaiSoNguyenTo(m,n):
    if (m >0 and n > 0):
        while (m != n):
            if (m > n):
                m = m - n
            else:
                n = n - m
        if (m == 1):
            print("Hai số này là nguyên tố cùng nhau")
        else:
            print("Hai số này không là nguyên tố cùng nhau")
```

Câu c).

```
def SecToHours(s):
    hour = s // 3600
    r = s%3600
    minute = r // 60
    second = r % 60
    print(s, "giây =", hour,"giờ",minute,"phút",second,"giây" )
```

Câu d).

```
def defineAbs(a, b):
    if a-b < 0:
        return b - a
    else:
        return a - b
```

4.16. Viết thủ tục NhapDL() có tính năng yêu cầu người dùng nhập số tự nhiên từ bàn phím. Chương trình sẽ yêu cầu nhập liên tục cho đến khi số cần nhập > 50 thì dừng lại.

```
1.  # 4.16
2.  def NhapDL():
3.      while True:
4.          n = int(input("Nhập số tự nhiên: "))
5.          if n > 50:
6.              return n
7.              break
8.  n = NhapDL()
9.  print("Số đã nhập:",n)
```

4.17. Viết hàm số f(m, n) với các tham số là các số tự nhiên. Hàm sẽ trả lại số nguyên tố lớn nhất đồng thời là ước chung của m và n. Nếu số đó không tồn tại thì trả lại 0.

Chú ý rằng vì 1 không được coi là số nguyên tố nên nếu p là số nguyên tố đồng thời là ước của m, n thì suy ra p > 1 và p phải là ước của d = USCLN(m, n). Vậy hàm f(m, n) sẽ phải được tính như sau:

- Nếu d = 1 (m, n nguyên tố cùng nhau) thì hàm f(m, n) = 0.

- Nếu d > 1 thì p sẽ là số nguyên tố lớn nhất là ước của d.

```
def f(m,n):
    while (m != n):
        if (m > n):
            m = m - n
        else:
            n = n - m
```

```
    if n == 1:
        return 0
    else:
        k = 2
        p = k
        while n > 1:
            while n%k != 0:
                k = k + 1
            p = k
            while n%k == 0:
                n = n//k
        return p
```

4.18. Viết hàm số factorial(n) tính giá trị là tích các số 1.2....n (gọi là n giai thừa).

Lời giải.

```
def factorial(n):
    p = 1
    for k in range(1,n+1):
        p = p*k
    return p
```

4.19. Viết hàm số nhuan(year) trả lại **True** nếu năm year là nhuận và **False** nếu year không là năm nhuận.

Lời giải.

```
def nhuan(year):
    if year % 400 == 0 or (year % 4 == 0 and year % 100 != 0):
        return True
    else:
        return False
```

4.20. Viết hàm số (hàm dấu) sign(x) trả lại 1 nếu x > 0, trả lại 0 nếu x = 0 và trả lại -1 nếu x < 0.

Lời giải.

```
def sign(x):
    if x > 0:
        return 1
    elif x==0:
        return 0
    else:
        return -1
```

Chương 5: Đối tượng

5.1. Đoạn chương trình sau khai báo một lớp và tạo một đối tượng cụ thể.

```
>>> class Hocsinh:
      Ten = "Hà"
>>> h = Hocsinh('Bình')
```

Hỏi kết quả đối tượng h được tạo ra có tên gì:

a) Có tên "Hà".

b) Có tên "Bình".

c) Không thể khởi tạo được.

Đáp án đúng: c).

5.2. Đoạn chương trình sau khai báo lớp và tạo đối tượng cụ thể.

```
>>> class Hocsinh:
      Ten = "Hà"
>>> h = Hocsinh()
>>> h.Ten = "Bình"
```

Hỏi kết quả đối tượng h được tạo ra có tên gì:

a) Có tên "Hà".

b) Có tên "Bình".

c) Không thể khởi tạo được.

Đáp án đúng: b).

5.3. Thiết lập lớp **QuadrilSquare** bao gồm đối tượng là các hình chữ nhật hoặc hình vuông.

- Mỗi đối tượng sẽ có 2 thuộc tính là chiều dài **w** và chiều cao **h**.

- Các method chính của lớp này là các hàm tính chu vi (Chuvi()) và diện tích (Dientich()) của mỗi đối tượng.

Lời giải.

```
class QuadrilSquare:
    def __init__(self, w, h):
        self.w = w
        self.h = h
    def Dientich(self):
        return self.w * self.h
    def Chuvi(self):
        return (self.w + self.h)*2
```

5.4. Thiết lập lớp **Hocsinh** dùng để quản lý các đối tượng học sinh trong lớp em.

Các thuộc tính của mỗi học sinh bao gồm:

- Họ và tên.

- Địa chỉ.

- Chiều cao.

- Cân nặng.

- Học lực.

Lớp Hocsinh được thiết lập với các phương thức sau:

- Hàm __init__() thiết lập tên học sinh, địa chỉ, chiều cao và cân nặng.

- Hàm cập nhật địa chỉ mới.

- Hàm cập nhật chiều cao, cân nặng mới.

- Hàm cập nhật học lực học sinh.

Lời giải.

```
class Hocsinh:
    def __init__(self, Name, address, height, weight, gpa):
        self.name = Name
        self.address = address
        self.height = height
        self.weight = weight
        self.gpa = gpa
    def UpdateAddress(self, address):
        self.address = address
    def UpdateWH(self, height, weight):
        self.height = height
        self.weight = weightUpdateGPA
    def UpdateGPA(self, gpa):
        self.gpa = gpa
```

5.5. Thiết lập lớp **Triple** bao gồm các đối tượng là một bộ 3 số thực (a, b, c). Thuộc tính của mỗi đối tượng bao gồm:

- Tên của đối tượng.

- Bộ 3 số a, b, c.

- Tổng 3 số này, ký hiệu S.

Thiết lập các phương thức sau với các đối tượng này:

- Hàm __init__() dùng để khởi tạo 1 bộ triple.

- Hàm Update() dùng để cập nhật giá trị S.

- Hàm Change() có chức năng thay đổi các giá trị a, b, c và đồng thời cập nhật S.

- Hàm isTriange() kiểm tra xem bộ 3 số này có tạo thành một tam giác hay không.

Lời giải.

```
class Triple:
    def __init__(self, a, b ,c):
        self.a = a
        self.b = b
        self.c = c
        self.S = self.a + self.b + self.c
    def Update(self):
        self.S = self.a + self.b + self.c
    def Change(self, a,b,c):
        self.a = a
```

```
        self.b = b
        self.c = c
        self.S = self.a + self.b + self.c
    def isTriangle(self):
        if self.a + self.b <= self.c or self.a + self.c <= self.b
or self.b + self.c <= self.a:
            return False
        else:
            return True
```

5.6. Thiết lập lớp **Tam_giac** là lớp con của lớp **Triple** trong bài tập 5. Bổ sung các phương thức sau cho lớp con này:

- Hàm tính diện tích tam giác.

- Hàm tính bán kính vòng tròn ngoại tiếp tam giác.

- Hàm tính bán kính vòng tròn nội tiếp tam giác.

Lớp con **Tam_giac** này sẽ kế thừa được phương thức nào từ lớp cha **Triple**?

Ta có các công thức sau tính diện tích S, bán kính vòng tròn ngoại tiếp R và bán kính vòng tròn nội tiếp r của tam giác có các cạnh a, b, c.

$S = \sqrt{p(p-a)(p-b)(p-c)}$ *với p = nửa chu vi tam giác.*

$R = \frac{abc}{4S}$

S = r.p hay r = S/p.

```
class Tam_Giac(Triple):
    def getpChuvi(self):
        return self.S/2
    def getArea(self):
        p = getpChuvi()
        area = (p*(p-self.a)*(p-self.b)*(p-self.c))**0.5
        return area
    def getCircumcircle(self):
        S = self.getArea()
        R = self.a*self.b*self.c/(4*S)
        return R
    def getInscribedCircle(self):
        S = self.getArea()
        p = self.getpChuvi()
        r = S/p
        return r
```

5.7. Có thể coi mỗi lớp là một kiểu dữ liệu mới được không? Giải thích lý do.

Trả lời: Có thể được. Mỗi lớp có thể tạo ra các đối tượng dữ liệu khác nhau nhưng cùng một kiểu dữ liệu mới.

5.8. Trong ví dụ lớp Tứ giác từ mục 5, giả sử chúng ta bổ sung thêm phương thức thay đổi tên của đối tượng tứ giác này. Hỏi khi đó các đối tượng của các lớp con hình bình hành, hình chữ nhật, hình vuông có kế thừa được phương thức này hay không?

Trả lời: Có.

5.9. Thiết kế 3 lớp kế thừa nhau là tam giác, tam giác cân và tam giác đều. Trong đó lớ tam giác cân là con của lớp tam giác, lớp tam giác đều là lớp con của lớp tam giác cân Thiết lập các phương thức tính chu vi, diện tích cho các đối tượng này.

Cần thiết lập 3 lớp kế thừa nhau như sau:

```
# 5.9
class Tamgiac:
    def __init__(self,a,b,c):
        self.a = a
        self.b = b
        self.c = c
        if not isTamgiac(a,b,c):
            print("3 cạnh không hợp lệ.")
    def isTamgiac(self):
        if self.a + self.b > self.c and self.b + self.c > self.a and self.a + self.c > self.b:
            return True
        else:
            return False
    def chuvi(self):
        return self.a + self.b + self.c
    def dientich(self):
        p = self.chuvi()/2
        return (p*(p-self.a)*(p-self.b)*(p-self.c))**0.5
class Tamgiac_can(Tamgiac):
    def __init__(self,a,b):
        self.a = a
        self.b = b
        self.c = b
class Tamgiac_deu(Tamgiac_can):
    def __init__(self,a):
        self.a = a
        self.b = a
        self.c = a
```

5.10. Lớp X được định nghĩa như sau:

```
class X(object):
    def id(self):
        print("I'm an X object")
```

Khi đó nếu viết các lệnh:

```
x = X()
x.id()
```

thì kết quả như thế nào?

a) "I'm an X object"

b) I'm an X object

c) Không thông báo gì.

Đáp án đúng: b).

5.11. Khi nào thì một thuộc tính của Lớp cha sẽ được kế thừa xuống cho Lớp con?

Trả lời: Khi thuộc tính này được khai báo và thiết lập trong lớp cha nhưng không được thiết lập lại trong lớp con.

5.12. Khi nào thì một phương thức của Lớp cha sẽ được kế thừa xuống cho Lớp con?

Trả lời: Một phương thức từ lớp cha sẽ được thừa kế xuống lớp con nếu nó không được định nghĩa lại trong lớp con này.

5.13. Thiết lập lớp Circle (vòng tròn) bao gồm thuộc tính R (bán kính) và tọa độ tâm X, Y. Thiết lập các phương thức tính chu vi và diện tích của đối tượng hình tròn.

Lời giải.

```
class Circle:
    def __init__(self,R,x,y):
        self.R = R
        self.x = x
        self.y = y
    def chuvi(self):
        pi = 3.14159
        return 2*self.R*pi
    def dientich(self):
        pi = 3.14159
        return pi*R*R
```

Chương 6: Kiểu dữ liệu List. Mảng một chiều

6.1. Lệnh list(range(-3,3)) trả lại giá trị nào?

a) [-3, -2, -1, 0, 1, 2, 3]

b) [-3, -2, -1, 0, 1, 2]

c) [-2, -1, 0, 1, 2]

d) [-3, 3]

Đáp án: b).

6.2. Kết quả đoạn chương trình sau là gì?

```
>>> A = []
>>> A.append([1,2,3])
>>> print(A)
```

a) [1, 2, 3]

b) 1, 2, 3

c) [[1, 2, 3]]

d) ([1, 2, 3])

Đáp án: c).

6.3. Kết quả của đoạn chương trình sau là gì?

```
>>> X = []
>>> X.append(1)
>>> X.append(5)
>>> print(X)
```

a) []

b) 1, 5

c) [1, 5]

d) 1 5

Đáp án: c).

6.4. Lệnh sau list(range(-3,3,3)) trả lại giá trị nào?

a) [-3, 3]

b) [-3, 0, 3]

c) [-3, 0]

Đáp án: c).

6.5. Lệnh sau list(range(-3)) trả lại giá trị nào?

a) [-3, 3]

b) [-3, 0, 3]

c) [-3, 0]

d) []

Đáp án: d).

6.6. Hàm số sau thực hiện công việc gì?

```
>>> def multi(start, stop):
      P = 1
      for i in range(start, stop):
            P = P * i
      return P
>>> multi(1,6)
```

Kết quả các lệnh trên là bao nhiêu?

a) 24

b) 100

c) 120

d) 720

Đáp án: c).

6.7. Kết quả của các lệnh sau như thế nào?

```
>>> A = [9, 1, 15, 0, "abc", 2, "de"]
>>> A.sort()
```

a) ["abc", "de", 0, 1, 2, 9, 15]

b) [0, 1, 2, 9, 15, "abc", "de"]

c) Thông báo lỗi.

d) [0, 1, 2, 9, 15]

Đáp án: c).

6.8. Viết chương trình nhập số tự nhiên n từ bàn phím, in và liệt kê danh sách các ước số thực sự của n như một danh sách (list).

Ví dụ nếu n = 15 thì kết quả dãy (list) là [1, 3, 5].

```
1.  # 6.8
2.  A = []
3.  n = int(input("Nhập 1 số tự nhiên n: "))
4.  for i in range(1, n):
5.      if n % i == 0:
6.          A.append(i)
7.  print(A)
```

6.9. Hàm số sau thực hiện công việc gì?

```
1.  def listover(A):
2.    B = []
3.    for item in range(len(A)-1,-1,-1):
4.      B.append(A[item])
5.    return B
```

Trả lời: Hàm số trả lại giá trị là dãy số ngược với A.

6.10. Viết chương trình thực hiện công việc sau:

- Nhập từ bàn phím số tự nhiên n.

- Nhập lần lượt n số tự nhiên tạo thành dãy $a_1, a_2, ..., a_n$. Giả sử dãy này gọi là A.

- Đưa ra dãy con $b_1, b_2, ..., b_k$ của dãy A bao gồm các phần tử là nguyên tố của dãy này

Cách làm sau thực hiện theo ý tưởng của sàng nguyên tố Eratosthenes.

```
# 6.10
A = []
B = []
n = int(input("Nhập số tự nhiên n: "))
for i in range(0, n):
    x = int(input("Nhập số thứ " + str(i+1)+ " :"))
    A.append(x)
max_val = max(A)
find_prime = [True for i in range(max_val + 1)]
find_prime[0] = False
find_prime[1] = False
k = 2
while k*k < max_val:
    if find_prime[k]:
        for j in range(k*k, max_val+1,k):
            find_prime[j] = False
    k += 1
lenn = len(A)
for k in range(lenn):
    if find_prime[A[k]]:
        B.append(A[k])
print(B)
```

6.11. Nhập dãy A bao gồm n phần tử từ bàn phím và tính tổng các phần tử của A.

```
# 6.11
n = int(input("Nhập số n: "))
A = []
for k in range(n):
x = float(input("Nhập số thứ " + str(k+1)+" : "))
A.append(x)
print("Các số đã nhập:",A)
print("Tổng:",sum(A))
```

6.12. Viết chương trình nhập số tự nhiên n từ bàn phím và tính các giá trị sau.

a) Tính tổng $1^2 + 2^2 + 3^2 + ... + n^2$.

b) Tính tổng $1 + 1/2 + 1/3 + ... + 1/n$.

c) Tính tổng $\frac{1}{2} + \frac{1}{6} + \frac{1}{12} + \cdots + \frac{1}{(n-1)n}$

Câu a).

```
# 6.12a
def SquareSum(n):
    ssum = 0
    for k in range(n+1):
        ssum = ssum + k*k
    return ssum
n = int(input("Nhập số n: "))
print(SquareSum(n))
```

Câu b)

```
# 6.12b
def SumPhanSo(n):
    sum1 = 0
    for i in range(1, n+1):
        sum1 = sum1 + 1/i
    return sum1
n = int(input("Nhập số n: "))
print(SumPhanSo(n))
```

Câu c).

```
# 6.12c
def SumPhanSo2(n):
    sum2 = 0
    for i in range(2, n+1):
        sum2 = sum2 + 1/((i-1)*i)
    return sum2
n = int(input("Nhập số n: "))
print(SumPhanSo2(n))
```

6.13. Cho trước 2 dãy A, B. Dãy C được thiết lập bằng cách lần lượt lấy các phần tử từ A, B và đưa vào C.

Ví dụ:

A = [0, 2, 4, 6, 8, 10].

B = [1, 3, 5, 7, 9, 11].

Thì C = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11].

Viết một hàm với 2 tham số đầu vào là A, B và trả lại dãy C như trên.

```
# 6.13
A = [0,2,4,6,8,10]
B = [1,3,5,7,9,11]
C = []
a = []
a.append(len(A))
a.append(len(B))
for i in range(0, max(a)):
    if i >= min(a):
        if len(A) == min(a):
            c.append(B[i])
            continue
        if len(B) == min(a):
            C.append(A[i])
            continue
    else:
            C.append(A[i])
            C.append(B[i])
print(C)
```

6.14. Cho trước dãy A. Dãy B được thiết lập bằng cách tách các phần tử là số (số nguyên hoặc số thực) từ dãy A. Ví dụ:

A = [1.2, "one" , 0.15, "A", "B", "C", 1, 50, 4.56, "C"]

Khi đó dãy B sẽ là:

B = [1.2, 0, 15, 1, 50, 4.56].

Viết một hàm số với tham số đầu vào là dãy A và kết quả trả lại dãy B.

Cách 1:

```
1. # 6.14
2. A = [1.2, "one" , 0.15, "A", "B", "C", 1, 50, 4.56, "C"]
3. def isNumber(obj):
4.     return type(obj) == type(1) or type(obj) == type(1ôt1)
5. B = []
6. for a in A:
7.     if isNumber(a):
8.         B.append(a)
9. print(B)
```

Cách 2: Sử dụng hàm có sẵn của Python isinstance().

```
1. # 6.14
2. A = [1.2, "one" , 0.15, "A", "B", "C", 1, 50, 4.56, "C"]
3. B = [s for s in A if isinstance(s, (int, float))]
4. print(B)
```

Chú ý: Hàm có sẵn isinstance() trong Python dùng để kiểm tra xem một đối tượng x r đó có phải là một phần tử thuộc một lớp nào đó, hoặc thuộc một trong các lớp nào hay không. Ví dụ:

- Kiểm tra xem x có phải là một xâu hay không?

```
>>> x = "abc"
>>> isinstance(x,str)
True
>>> x = 1.2
>>> isinstance(x,str)
False
```

- Kiểm ra xem a có phải là số nguyên hay thực hay không?

```
>>> a = 10
>>> isinstance(a,(int,float))
True
```

6.15. Viết chương trình nhập một xâu ký tự từ bàn phím. Kết quả là một dãy các ký của xâu này. Ví dụ:

Xâu nhập vào: Việt Nam.

Kết quả ra: ["V", "i", "ệ", "t", " ", "N", "a", "m"].

```
1. # 6.15
2. s = input("Nhập 1 xâu: ")
3. li = list(s)
4. print(li)
```

6.16. Viết chương trình nhập số tự nhiên n và thiết lập:

- Dãy A bao gồm các số tự nhiên lẻ nhỏ hơn n.

- Dãy B bao gồm các số tự nhiên chẵn nhỏ hơn n.

```
1. # 6.16
2. A = []
```

```
3.  B = []
4.  n = int(input("Nhập số n: "))
5.  for i in range(n):
6.      if i % 2 == 0:
7.          A.append(i)
8.      else:
9.          B.append(i)
10. print(A)
11. print(B)
```

6.17. Cho trước 2 dãy số A, B. Dãy C được tạo ra theo quy tắc sau:

Giả sử số phần tử của A là n và nhỏ hơn số phần tử của B. Trường hợp ngược lại làm tương tự.

- n phần tử đầu tiên của C là tổng các số hạng tương ứng của A, B.

- Các phần tử còn lại lấy từ B.

Viết thủ tục với 2 tham số đầu vào là A, B và in ra màn hình dãy C.

```
1.  # 6.17
2.  # Chú ý: dùng hàm randint() để sinh bộ dữ liệu ngẫu nhiên.
3.  def SinhDL():
4.      import random
5.      A = []
6.      B = []
7.      n = int(input("Nhập n nhỏ hơn hoặc lớn hơn bằng 6: "))
8.      for i in range(0, n):
9.          n = random.randint(5, 30)
10.         A.append(n)
11.     for j in range(0, 6):
12.         n = random.randint(5, 30)
13.         B.append(n)
14.     return A, B
15.
16. def SinhC(A,B):
17.     point = 0 # Bắt đầu cộng
18.     if (len(B) > len(A)):
19.         C = B.copy()
20.         for a in A:
21.             C[point] = a + C[point]
22.             point += 1
23.     else:
24.         C = A.copy()
25.         for b in B:
26.             C[point] = b + C[point]
27.             point += 1
28.     return C
29.
30. A,B = SinhDL()
31. C = SinhC(A,B)
32. print(A)
33. print(B)
34. print(C)
```

Kết quả chạy chương trình có thể như sau:

```
Nhập n nhỏ hơn hoặc lớn hơn bằng 6: 10
[29, 29, 29, 15, 30, 23, 7, 28, 24, 18]
[30, 9, 15, 9, 14, 22]
[59, 38, 44, 24, 44, 45, 7, 28, 24, 18]
```

6.18. Viết các hàm số với đầu vào là dãy A (bao gồm số và xâu ký tự).

a) Tính tổng các phần tử là số trong dãy A.

b) Đếm số các phần tử là xâu ký tự từ dãy A.

c) Đưa ra kết quả là 2 dãy B, C. Dãy B bao gồm các phần tử là số của A, dãy C bao gồ các phần tử là xâu ký tự của A.

Để làm được bài tập này chúng ta cần thiết lập một số hàm sau:

- Hàm kiểm tra một đối tượng có là kiểu số (số nguyên hoặc số thực) hay không.

```
def isNumber(obj):
    if type(obj) == type(1) or type(obj) == type(1.1):
        return True
    else:
        return False
```

- Hàm kiểm tra một đối tượng có là kiểu str hay không.

```
def isString(obj):
    if type(obj) == type(""):
        return True
    else:
        return False
```

Câu a).

Hàm tính tổng các số hạng là số trong dãy A như sau:

```
def sumList(A):
    S = 0
    for x in A:
        if isNumber(x):
            S = S + x
    return S
```

Câu b).

Hàm đếm số các phần tử là xâu trong dãy A như sau:

```
def countString(A):
    count = 0
    for s in A:
        if isString(s):
            count = count + 1
    return count
```

Câu c)

Hàm sau tách từ dãy A thành 2 dãy B, C. B gồm các phần tử là số và C bao gồm c phần tử là xâu của dãy A.

```
def filterList(A):
    B = []
    C = []
    for x in A:
        if isNumber(x):
```

```
            B.append(x)
        if isString(x):
            C.append(x)
    return B, C
```

6.19. Viết một hàm số thay thế cho phương thức **sort()** của dãy (list). Tham số đầu vào là dãy A bao gồm các phần tử là số (nguyên hoặc thực). Hàm trả lại là một dãy bao gồm các số đã được sắp xếp theo thứ tự tăng dần các phần tử từ A.

```
1.  # 6.19
2.  A = [1,0,1.2, 4.5, 7, -1, 10, 7, 9, 1, 20, 15]
3.  n = len(A)
4.  def sort(A):
5.      B = A.copy()
6.      for i in range(n-1):
7.          for j in range(i+1,n):
8.              if B[i] > B[j]:
9.                  B[i],B[j] = B[j],B[i]
10.     return B
11. B = sort(A)
12. print(A)
13. print(B)
```

6.20. Viết một hàm số tổng quát hơn phương thức **sort()** của dãy (list) như sau:

- Đầu vào là dãy A bao gồm có thể là số hoặc ký tự.

- Đầu ra là dãy B thu được từ dãy A bằng cách sắp xếp các phần tử là số theo thứ tự tăng dần, còn các vị trí là ký tự thì giữ nguyên.

Ví dụ:

A = [1.2, "one", 0, 15, "A", "B", "C", 1, 50, 4.56, "C"].

B = [0, "one", 1, 1.2, "A", "B", "C", 4.56, 15, 50, "C"].

Đặt tên hàm số này là **general_sort()**.

*Hàm sắp xếp **general_sort()** hoàn toàn tương tự hàm **sort()** của bài trên, điểm khác biệt là các phép so sánh chỉ được thực hiện với các số hạng kiểu số (số nguyên hoặc số thực). Hàm **isNumber(obj)** sẽ kiểm tra xem đối tượng obj có phải là số hay không.*

```
1.  # 6.20
2.  A = [1.2, "one", 0, 15, "A", "B", "C", 1, 50, 4.56, "C"]
3.  n = len(A)
4.  def isNumber(obj):
5.      if type(obj) == type(1) or type(obj) == type(1.1):
6.          return True
7.      else:
8.          return False
9.  def general_sort(A):
10.     B = A.copy()
11.     for i in range(n-1):
12.         for j in range(i+1,n):
13.             if isNumber(B[i]) and isNumber(B[j]) and B[i] >
    B[j]:
14.                 B[i],B[j] = B[j],B[i]
15.     return B
16. B = general_sort(A)
```

```
print(A)
print(B)
```

6.21. Cho trước 2 dãy (list) A, B và số tự nhiên k, $0 \le k < len(A)$. Viết hàm số trả lại dã (list) thu được từ A sau khi chèn dãy B vào vị trí của chỉ số k. Ví dụ:

A = [0, 1, 2, 3, 4, 5], B = ["a", "b", "c"], k = 1 thì dãy (list) thu được sẽ là:

C = [0, "a", "b", "c", 1, 2, 3, 4, 5].

*Hàm chính của chương trình là **insert(A, B, k)** có chức năng chèn dãy B vào dãy A t vị trí k.*

```
# 6.21
A = [0, 1, 2, 3, 4, 5]
B = B = ["a","b","c"]
k = 1
def insert(A,B,k):
    if k < 0:
        C = B + A
    elif k >= len(A):
        C = A + B
    else:
        C = A[ : k] + B + A[k : ]
    return C
C = insert(A,B,k)
print(C)
```

6.22. Cho trước dãy (list) a bao gồm các số nguyên, cho trước số nguyên k. Biết rằng d (list) trong Python có 2 phương thức là **count()** và **remove()** có ý nghĩa như sau:

a.count(<giá trị>) sẽ trả lại số lần tìm thấy của <giá trị> trong a, nếu không tìm thấy trả lại 0.

a.remove(<giá trị>) sẽ xóa phần tử đầu tiên có <giá trị> trong a, nếu không tìm thấy ph tử có <giá trị> sẽ báo lỗi.

Viết chương trình xóa tất cả các phần tử trong dãy a có giá trị k. Chú ý viết chương trì không được để gặp lỗi.

```
# 6.22
a = [1,0,1,1,0,2,3,4,0,1,2,0,1,2,6,7.8,9,0,0,1,1,10]
def removeALL(a,k):
    b = a.copy()
    while b.count(k) > 0:
        b.remove(k)
    return b
print("Dãy a gốc:")
print(a)
k = int(input("Nhập giá trị k: "))
b = removeALL(a,k)
print("Dãy đã thay đổi:")
print(b)
```

6.23. Cho trước dãy (list) a bao gồm các số nguyên, cho trước 2 số nguyên k, m. V chương trình thay thế tất cả các phần tử của a có giá trị k bằng giá trị m. Gợi ý: Sử dụ phương thức **index()** của list trong Python.

```
# 6.23
a = [1,0,1,1,0,2,3,4,0,1,2,0,1,2,6,7.8,9,0,0,1,1,10]
def change(a,k,m):
    b = a.copy()
    for i in range(len(b)):
        if b[i] == k:
            b[i] = m
    return b
print("Dãy a gốc:")
print(a)
k = int(input("Nhập giá trị k: "))
m = int(input("Nhập giá trị m: "))
b = change(a,k,m)
print("Dãy đã thay đổi:")
print(b)
```

6.24. Cho trước 2 dãy (list) a, b. Viết chương trình thiết lập dãy c bao gồm các phần tử lấy lần lượt từ phần cuối của a, b. Gợi ý: Sử dụng phương thức **pop()** của list trong Python.

```
# 6.24
a = [0,2,4,6,8,10]
b = [1,3,5,7,9,11,13,15]
c = []
findmax = [len(a),len(b)]
for i in range(max(findmax)):
    if i >= min(findmax):
        if len(b) == 0 and len(a) != 0:
            c.append(a.pop())
        else:
            c.append(b.pop())
    else:
        c.append(a.pop())
        c.append(b.pop())
print(c)
```

Kết quả chạy chương trình trên với ví dụ cụ thể của các dãy a, b như trên ta có:

```
[10, 15, 8, 13, 6, 11, 4, 9, 2, 7, 0, 5, 3, 1]
```

6.25. Hãy viết một lệnh để tạo dãy a bao gồm m số 0 và n số 1 với m, n là các số tự nhiên cho trước.

Ví dụ một lời giải có thể như sau:

```
a =[0]*m + [1]*n
```

6.26. Cho trước dãy (list) A. Lệnh sau thực hiện công việc gì (với i, j là các chỉ số bất kỳ)?

```
A[i], A[j] = A[j], A[i]
```

Trả lời: Lệnh này thực hiện công việc đổi vị trí 2 phần tử A[i], A[j] của dãy A.

6.27. Cho trước dãy (list) A. Viết chương trình tạo ra dãy B bao gồm các phần tử lấy từ A nhưng theo thứ tự ngược lại. Chú ý yêu cầu giữ nguyên dãy A không thay đổi.

Có nhiều cách làm bài tập này. Ví dụ:

Cách 1:

```
B = A[ :  : -1]
```

Cách 2:

```
B = A[-1 : -len(A)-1: -1]
```

Cách 3:

```
B = A.copy()
B.reverse()
```

6.28. Cho trước 2 dãy số A, B có số phần tử bằng nhau và đều là các số tự nhiên chương trình tạo dãy C cũng có số phần tử bằng A, B và các phần tử của C là u chung lớn nhất của các phần tử tương ứng của A, B.

```
1.  # 6.28
2.  A = [30,70,4,6,8,10]
3.  B = [6,12,5,7,9,11]
4.  C = []
5.  def gcd(x,y):
6.      while (y != x):
7.          if x > y:
8.              x = x - y
9.          else:
10.             y = y - x
11.     return x
12. for i in range(len(A)):
13.     C.append(gcd(A[i],B[i]))
14. print(C)
```

Chương 7: List của List. Mảng nhiều chiều

7.1. Đoạn chương trình sau in ra kết quả nào?

```
Xlist = [1, [1, 2], [1, 2, 3], [1, 2, 3, 4]]
print(Xlist[2] + [1])
```

a) 4

b) [1, 2, 3, 1]

c) [[1, 2], 1]

d) [[1, 2, 3], [1, 2, 3, 1]]

Đáp án: b.

7.2. Kết quả đoạn chương trình sau là bao nhiêu?

```
X = [[1], [1, 2], [1, 2, 3], [1, 2, 3, 4]]
sum = 0
for j in range(len(X)):
    for i in range(len(X[j])):
        sum = sum + X[j][i]
print(sum)
```

a) 0

b) 10

c) 20

d) 50

Đáp án: c.

7.3. Kết quả đoạn chương trình sau như thế nào?

```
for i in range(4):
    for j in range(3):
      print(i * j, end = "")
```

a) 0123456789101112

b) 00001111222233334444

c) 0000124246

d) 000012024036

Đáp án: d.

7.4. Đoạn chương trình sau kết quả thể hiện như thế nào?

```
>>> x = [1]
>>> y = x
>>> x = x + [2]
>>> print(x, y)
```

Hãy giải thích kết quả trên.

Lời giải.

Kết quả của chương trình trên sẽ in ra x, y như sau:

```
[1, 2] [1]
```

Tức là x = [1, 2], y = [1]

Sau lệnh thứ 2 thì x và y cùng trỏ vào đối tượng [1]. Nhưng lệnh x = x + [2] đã tạo ra m đối tượng x mới là [1, 2] và khác biệt với y. Do vậy kết quả x, y sẽ trỏ vào 2 đối tượr khác nhau.

7.5. Đoạn chương trình sau kết quả thể hiện như thế nào?

```
>>> x = [1]
>>> y = x
>>> x[0] = 15
>>> print(x, y)
```

Hãy giải thích kết quả trên.

Lời giải.

Kết quả các lệnh trên thu được như sau:

```
[15] [15]
```

Vậy x và y cùng trỏ vào một đối tượng [15].

Sau lệnh thứ 2, y = x thì x và y cùng trỏ vào đối tượng là [1]. Lệnh cuối cùng x[0] = không làm thay đổi đối tượng x nên sau lệnh trên x, y vẫn trỏ vào một đối tượng là [1

7.6. Kết quả của các lệnh sau là gì?

```
>>> s = "cable"
>>> s[3 : -6: -1]
```

a) 'elbac'

b) 'lbac'

c) 'lbace'

d) 'bacel'

Đáp án: b.

7.7. Sau khi chạy chương trình sau, các dãy B, C sẽ như thế nào?

```
B = C = []
for i in range(3):
  B.append(i)
for i in range(4):
  C.append(i)
```

a) B = [0, 1, 2], C = [0, 1, 2, 3].

b) B = [0, 1, 2, 0, 1, 2, 3], C = [0, 1, 2, 3].

c) B = [0, 1, 2], C = [0, 1, 2, 0, 1, 2, 3].

d) B = [0, 1, 2, 0, 1, 2, 3], C = [0, 1, 2, 0, 1, 2, 3].

Đáp án: d.

7.8. Ma trận m × n (m hàng, n cột) có thể được mô tả bởi danh sách như sau:

A = [[a_{11}, a_{12}, …, a_{1n}], [a_{21}, a_{22}, …, a_{2n}], …., [a_{m1}, a_{m2}, …, a_{mn}]]

Cho trước ma trận (hay list) A, viết chương trình tính:

- Số hàng của ma trận trên.

- Số cột của ma trận trên.

Gợi ý:

Số hàng của ma trận A bằng len(A).

Số cột của ma trận A bằng len(A[0]).

7.9. Giả sử A là dãy mô tả ma trận m × n như bài trên. Ví dụ:

A = [[1, 0, 5], [13, -1, 0], [2, 16, -18], [10, 9, -1]]

Hỏi sau khi chạy chương trình sau, các dãy B, C thể hiện thông tin gì?

```
B = []
C = []
for item in A:
    sum = 0
    for i in item:
        sum = sum + i
    B.append(sum)
n = len(A[0])
for j in range(n):
    sum = 0
    for item in A:
        sum = sum + item[j]
    C.append(sum)
```

Trả lời:

Dãy B có m phần tử, phần tử thứ k của B là tổng các phần tử của hàng thứ k của A, tức là tổng các số của dãy A[k].

Dãy C có n phần tử. Phần tử thứ k của C là tổng các số cột k của A.

7.10. Giả sử A là list mô tả ma trận m × n như các bài tập trên.

A = [[a_{11}, a_{12}, ..., a_{1n}], [a_{21}, a_{22}, ..., a_{2n}], ..., [a_{m1}, a_{m2}, ..., a_{mn}]]

Viết thủ tục **ChangeRow(A,i,j)** có chức năng đổi vị trí 2 hàng i, j của ma trận A.

Thủ tục này có thể viết đơn giản như sau:

```
def ChangeRow(A,i,j):
    temp = A[i].copy()
    A[i] = A[j].copy()
    A[j] = temp
```

7.11. Giả sử A là list mô tả ma trận m × n như các bài tập trên.

A = [[a_{11}, a_{12}, ..., a_{1n}], [a_{21}, a_{22}, ..., a_{2n}], ..., [a_{m1}, a_{m2}, ..., a_{mn}]]

Viết thủ tục **ChangeColumn(A,i,j)** có chức năng đổi vị trí 2 cột i, j của ma trận A.

Thủ tục này có thể viết như sau:

```
def ChangeColumn(A,i,j):
    for k in range(len(A)):
        A[k][i],A[k][j] = A[k][j], A[k][i]
```

7.12. Cho 2 list biểu diễn ma trận A, B cùng kích thước. Viết thủ tục **Sum(A, B)** trả lại list C là tổng của 2 ma trận A, B.

Thủ tục tính tổng 2 ma trận cùng kích thước có thể như sau:

```
def Sum(A,B):
    m = len(A)
    n = len(A[0])
    C = []
    for i in range(m):
        temp = [0]*n
        C.append(temp)
    for i in range(m):
        for j in range(n):
            C[i][j] = A[i][j] + B[i][j]
    return C
```

7.13. Cho 2 list biểu diễn ma trận A, B cùng kích thước. Viết thủ tục **Diff(A, B)** trả lại l C là hiệu của 2 ma trận A, B.

Lời giải bài tập này tương tự bài 7.12.

7.14. Cho dãy A biểu diễn ma trận kích thước m × n.

Viết thủ tục **Scalar(A,k)** trả lại list biểu diễn ma trận C = kA, phép nhân ma trận A v số thực k.

Hàm Scalar(A,k) có thể như sau:

```
def Scalar(A,k):
    m = len(A)
    n = len(A[0])
    C = A.copy()
    for i in range(m):
        for j in range(n):
            C[i][j] = A[i][j]*k
    return C
```

7.15. Cho 2 ma trận A, B kích thước m × n và n × p.

Viết thủ tục **Multiple(A, B)** trả lại list là ma trận C là tích của 2 ma trận A, B.

Theo định nghĩa ma trận tích C = A × B có kích thước m × p và được định nghĩa như sa

$$C_{ij} = \sum_{k=1}^{n} A_{ik} . B_{kj}$$

Chú ý: *Hàm* ***Multiple(A, B)*** *sẽ phụ thuộc vào hàm* ***NewMatrix(m, n)*** *có chức năng s một ma trận C kích thước m × n và bao gồm toàn số 0*

```
1.  # 7.15
2.  def newMatrix(m,n):
3.      C = []
4.      for i in range(m):
5.          temp = [0]*n
6.          C.append(temp)
7.      return C
8.  def Multiple(A,B):
9.      m = len(A)
10.     n = len(A[0])
11.     p = len(B[0])
12.     C = newMatrix(m,p)
```

```
14.             for j in range(p):
15.                 S = 0
16.                 for k in range(n):
17.                     S = S + A[i][k]*B[k][j]
18.                 C[i][j] = S
19.     return C
```

7.16. Chứng minh rằng với dãy (list, tuple, string) có độ dài **len** thì quan hệ giữa chỉ số dương và chỉ số âm được cho trong bảng sau:

Chỉ số dương	Chỉ số âm
p	p - len
q + len	q

Lời giải.

Chúng ta chứng minh các mệnh đề của từng hàng.

- Theo định nghĩa chỉ số âm thì phần tử của dãy với chỉ số dương len-1 sẽ tương ứng với chỉ số âm -1. Vậy suy ra phần tử với chỉ số dương len – 1 – k sẽ tương ứng chỉ số âm -1 – k, vậy nếu đặt len – 1 – k = p thì suy ra -1 – k = p – len. Vậy mệnh đề của dòng đầu tiên được chứng minh.

- Với dòng thứ 2, nếu đặt p = len + q thì theo chứng minh trên (hàng 1) ta có chỉ số âm tương ứng sẽ là p – len = len + q – len = q. Đó là đpcm.

7.17. Cho trước dãy số bất kỳ:

$a_1, a_2, ..., a_N$ (*)

Các phần tử của dãy có thể được nhập từ bàn phím hoặc cho trước trong chương trình.

Viết các chương trình thực hiện các công việc sau:

a) Kiểm tra xem dãy (*) có phải là đơn điệu tăng hay không?

b) Kiểm tra xem dãy (*) có phải là đơn điệu tăng thực sự hay không?

c) Kiểm tra xem dãy (*) có phải là đơn điệu giảm hay không?

d) Kiểm tra xem dãy (*) có phải là đơn điệu giảm thực sự hay không?

Câu a.

```
1.  # 7.17a
2.  A = [1,2,3,4,5,6,7,8,9,10,11,12,100,99]
3.  def kt_Increase(A):
4.      n = len(A)
5.      kq = True
6.      for i in range(n-1):
7.          if A[i] > A[i+1]:
8.              kq = False
9.              break
10.     return kq
11. if kt_Increase(A):
12.     print("Đơn điệu tăng.")
13. else:
14.     print("Không đơn điệu tăng.")
```

Câu b.

Lời giải giống bài 7.17a, chỉ khác tại dòng lệnh 7, thay thế > bằng >=.

Câu c.

Lời giải giống bài 7.17a, chỉ khác tại dòng lệnh 7, thay thế > bằng <.

Câu d.

Lời giải giống bài 7.17a, chỉ khác tại dòng lệnh 7, thay thế > bằng <=.

7.18. Cho trước dãy số bất kỳ:

$a_1, a_2, ..., a_N$ (*)

Các phần tử của dãy có thể được nhập từ bàn phím hoặc cho trước trong chương trình

Viết chương trình tìm một dãy con liên tục đơn điệu tăng lớn nhất trong dãy trên.

Chú ý: Chỉ cần chỉ ra một dãy con như vậy.

Ví dụ nếu dãy ban đầu là:

3 1 2 4 -1 8 7 10 11 7 8 9 0 16 40 50 -100

Dãy con liên tục đơn điệu tăng dài nhất là 0 16 40 50.

```
1.  # Bài 7.18
2.  A = [3,1,2,4,-1,8,7,10,11,7,8,9,0,16,40,50,-100]
3.  imax = 0
4.  lenmax = 1
5.  icurr = 0
6.  lencurr = 1
7.  for k in range(len(A)-1):
8.      if A[k] < A[k+1]:
9.          lencurr = lencurr + 1
10.     else:
11.         if lenmax < lencurr:
12.             imax = icurr
13.             lenmax = lencurr
14.         icurr = k+1
15.         lencurr = 1
16. print(A)
17. print(A[imax:imax + lenmax])
```

Kết quả chạy chương trình trên như sau:

```
[3, 1, 2, 4, -1, 8, 7, 10, 11, 7, 8, 9, 0, 16, 40, 50, -100]
[0, 16, 40, 50]
```

7.19. Cho trước một dãy số chỉ bao gồm các số 0 và 1.

Viết chương trình chỉ ra vị trí và độ dài của một dãy con liên tục dài nhất toàn các số

Lời giải.

Bài tập này có thuật toán tương tự bài 7.18 là chỉ cần một vòng lặp, tuy nhiên có nhiều sự khác biệt.

```
1.  # Bài 7.19
2.  A = [0,1,0,0,0,1,1,0,0,1,0,0,0,0,0,1,1,0,1,0,1,1,1,1,1,0,0,0]
3.  imax = -1
4.  lenmax = 0
```

```
icurr = -1
lencurr = 0
for k in range(len(A)-1):
    if A[k] == 0 and A[k+1] == 0:
        lencurr = lencurr + 1
    elif A[k] == 0 and A[k+1] == 1:
        if lenmax < lencurr:
            imax = icurr
            lenmax = lencurr
    elif A[k] == 1 and A[k+1] == 0:
        if imax == -1:
            imax = k+1
            lenmax = 1
        icurr = k+1
        lencurr = 1
    else:
        continue
print(A)
print(imax,lenmax)
print(A[imax:imax + lenmax])
```

Kết quả chạy chương trình:

```
[0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1,
1, 1, 1, 0, 0, 0]
10 5
[0, 0, 0, 0, 0]
```

7.20. Giả sử A là list mô tả ma trận m × n như trong bài tập 8.

A = [[a_{11}, a_{12}, ..., a_{1n}], [a_{21}, a_{22}, ..., a_{2n}], ..., [a_{m1}, a_{m2}, ..., a_{mn}]]

Ma trận A được mô tả như list trên chính là ma trận sau:

$$A = \begin{pmatrix} a_{11}a_{12} \dots a_{1n} \\ a_{21}a_{22} \dots a_{2n} \\ a_{m1}a_{m2} \dots a_{mn} \end{pmatrix}$$

Phép lấy đối xứng của ma trận A là ma trận sau:

$$A^T = \begin{pmatrix} a_{11}a_{21} \dots a_{m1} \\ a_{12}a_{22} \dots a_{m2} \\ a_{1n}a_{2n} \dots a_{mn} \end{pmatrix}$$

Viết chương trình cho trước ma trận A và tính ma trận A^T.

```
# 7.20
A = [[1,2,3,4],[5,6,7,8],[9,10,11,12]]
def Symmetric(A):
    m = len(A)
    n = len(A[0])
    C = []
    for j in range(n):
        temp = []
        for i in range(m):
            temp.append(A[i][j])
        C.append(temp)
    return C
def showMatrix(A):
```

```
14.       m = len(A)
15.       n = len(A[0])
16.       for i in range(m):
17.           for j in range(n):
18.               print(A[i][j],end = " ")
19.           print()
20. C = Symmetric(A)
21. print("Ma trận A:")
22. showMatrix(A)
23. print("Ma trận AT:")
24. showMatrix(C)
```

7.21. Mô tả ý nghĩa và kết quả thực hiện các lệnh sau từ Python Shell.

```
>>> G = (2*i + 1 for i in range(10))
>>> for i in G:
      print(i)
      if i == 9:
          break
>>> for k in G:
      print(k, end = " ")
```

Trả lời:

- Lệnh thứ nhất tạo một generator G có thể sinh 10 số tự nhiên lẻ bắt đầu từ 1.

- Lệnh thứ 2 sẽ in ra 9 số lẻ đầu tiên của G.

- Lệnh thứ 3 sẽ in ra số lẻ cuối cùng của G.

7.22. Viết hàm sinh dữ liệu tuần tự theo yêu cầu sau:

a) Hàm sinh n số chẵn đầu tiên tính từ 0.

b) Hàm sinh n số chính phương đầu tiên.

Câu a.

```
# 7.22a
def even(n):
    for k in range(0,2*n,2):
        yield k
```

Câu b.

```
# 7.22b
def chinhphuong(n):
    for k in range(1,n+1):
        yield k*k
```

7.23. Viết hàm GenPrime(n) sinh generator bao gồm n số nguyên tố đầu tiên.

*Với bài tập này cách làm nhanh nhất là viết riêng một hàm kiểm tra số nguyên tố lập, sau đó viết hàm **GenPrime(n)**, trong đó sử dụng hàm kiểm tra kia. Ở đây trình lời giải khác viết gộp chung vào một hàm duy nhất.*

```
1.  # Hàm Genprime
2.  def GenPrime(n):
3.      m = 2
4.      k = 1
5.      while k <= n:
6.          count = 0
7.          for j in range(1,m):
```

```
8.                  if m%j == 0:
9.                      count = count + 1
10.             if count == 1:
11.                 yield m
12.                 k = k + 1
13.         m = m + 1
```

Đoạn chương trình sau sử dụng hàm sinh GenPrime(100) sẽ in ra 100 số nguyên tố đầu tiên của dãy số tự nhiên.

```
for p in GenPrime(100):
    print(p, end = " ")
```

Kết quả như sau:

```
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83
89 97 101 103 107 109 113 127 131 137 139 149 151 157 163 167 173
179 181 191 193 197 199 211 223 227 229 233 239 241 251 257 263
269 271 277 281 283 293 307 311 313 317 331 337 347 349 353 359
367 373 379 383 389 397 401 409 419 421 431 433 439 443 449 457
461 463 467 479 487 491 499 503 509 521 523 541
```

7.24. Nếu trong bài tập 7.21 thay G = range(1, 20, 2) thì kết quả có thay đổi gì không?

Trả lời:

Nếu thay G = range(1,20,2) trong bài tập 7.21 thì:

- Lệnh thứ 2 sẽ vẫn in ra 9 số lẻ đầu tiên của G.

- Lệnh thứ 3 sẽ in ra đủ 10 số lẻ của G.

7.25. Sử dụng chức năng **list comprehension** để tạo các dãy (list) sau:

a) Dãy tất cả các số là bội của 3 và nhỏ hơn 100.

b) Dãy 20 số chính phương đầu tiên.

c) Dãy 10 xâu nhị phân theo quy luật: 1, 10, 100, ...

d) Dãy 10 xâu nhị phân theo quy luật: 100000000, 0100000000, ..., 0000000001.

Câu a.

```
A = [n for n in range(100) if n%3==0]
```

Câu b.

```
A = [k*k for k in range(1,21)]
```

Câu c.

```
A = ["1"+"0"*k for k in range(10)]
```

Câu d.

```
A = ["0"*k + "1" + "0"*(9-k) for k in range(10)]
```

7.26. Viết chương trình sinh dãy (list) A là biểu diễn của ma trận đơn vị.

Thủ tục sinh ma trận đơn vị bậc n như sau:

```
def UnitMatrix(n):
    A = []
    for i in range(n):
        temp = [0]*i + [1] + [0]*(n-i)
        A.append(temp)
    return A
```

Chương trình nhập số n từ bàn phím và sinh ma trận đơn vị bậc n, sau đó hiện kết q trên màn hình.

```
# 7.26
# Sinh ma trận vuông đơn vị bậc n
n = int(input("Nhập số tự nhiên n: "))
def UnitMatrix(n):
    A = []
    for i in range(n):
        temp = [0]*i + [1] + [0]*(n-i)
        A.append(temp)
    return A
def showMatrix(A):
    n = len(A)
    for i in range(n):
        for j in range(n):
            print(A[i][j], end = " ")
        print()
A = UnitMatrix(n)
showMatrix(A)
```

7.27. Viết hàm số newMatrix(n) sinh ma trận vuông n×n bao gồm toàn các số 0.

Lời giải.

```
def newMatrix(n):
    A = []
    for i in range(n):
        A.append([0]*n)
    return A
```

7.28. Viết hàm số newMatrix(m, n) sinh ma trận kích thước m×n bao gồm toàn các số

Lời giải.

```
def newMatrix(m,n):
    A = []
    for i in range(m):
        A.append([0]*n)
    return A
```

Chương 8: Khái niệm Module

8.1. Lệnh:

```
>>> abs(1 + 1j)
```

trả lại giá trị bao nhiêu?

a) 1

b) 2

c) 1.41421356

d) Lệnh báo lỗi.

Đáp án đúng: c).

8.2. Lệnh:

```
>>>import cmath
>>> (1.5 + j).real
```

trả lại giá trị nào?

a) 1

b) 1.5

c) 0

d) Lệnh báo lỗi

Đáp án đúng: d).

Chú ý: Câu lệnh trên cần viết đúng là:

```
(1.5 + 1j).real
```

8.3. Lệnh:

```
>>>import cmath
>>> (2 + 3j).imag
```

trả lại giá trị nào?

a) 2

b) 3

c) 5

d) Lệnh báo lỗi.

Đáp án đúng: b).

8.4. Viết chương trình sinh một dãy (list) A bao gồm 1000 số tự nhiên ngẫu nhiên nằm trong khoảng [1, 99999].

```
1.  # 8.4
2.  def genList(m,n):
3.      from random import randint
4.      A = []
5.      for i in range(1000):
6.          A.append(randint(m,n))
7.      return A
```

```
8. A = genList(1,99999)
9. print(A)
```

8.5. Viết chương trình sinh một dãy (list) A bao gồm 1000 số tự nhiên ngẫu nhiên n trong khoảng [1, 99999], sau đó sắp xếp lại dãy này theo thứ tự tăng dần.

```
1.  # 8.5
2.  def genList(m,n):
3.      from random import randint
4.      A = []
5.      for i in range(1000):
6.          A.append(randint(m,n))
7.      return A
8.  A = genList(1,99999)
9.  A.sort()
10. print(A)
```

8.6. Công thức tính diện tích tam giác có các cạnh a, b, c là:

$$S = \sqrt{p(p-a)(p-b)(p-c)}.$$

Ở đây p = nửa chu vi tam giác.

Viết hàm số tính diện tích tam giác với các tham số đầu vào là a, b, c - số đo các cạnh c tam giác.

```
1.  # 8.6
2.  def isTamgiac(a,b,c):
3.      if a <= 0 or b <= 0 or c <= 0:
4.          return False
5.      else:
6.          return a + b > c and b + c > a and c + a > b
7.  def Dientich(a,b,c):
8.      from math import sqrt
9.      if isTamgiac(a,b,c):
10.         p = (a + b + c)/2
11.         return sqrt(p*(p-a)*(p-b)*(p-c))
12.     else:
13.         return -1
14. S = Dientich(1,1,1)
15. print(S)
```

8.7. Viết chương trình nhập tọa độ 3 đỉnh A, B, C và tính chu vi, diện tích tam giác AI

```
1.  # 8.7
2.  def isTamgiac(a,b,c):
3.      if a <= 0 or b <= 0 or c <= 0:
4.          return False
5.      else:
6.          return a + b > c and b + c > a and c + a > b
7.  def Dientich(a,b,c):
8.      from math import sqrt
9.      if isTamgiac(a,b,c):
10.         p = (a + b + c)/2
11.         return sqrt(p*(p-a)*(p-b)*(p-c))
12.     else:
13.         return -1
14. def dlength(x1,y1,x2,y2):
```

```
15.     from math import sqrt
16.     return sqrt((x1-x2)*(x1-x2) + (y1-y2)*(y1-y2))
17. def tinhS(pA,pB,pC):
18.     a = dlength(pB[0],pC[0],pB[1],pC[1])
19.     b = dlength(pA[0],pC[0],pA[1],pC[1])
20.     c = dlength(pB[0],pA[0],pB[1],pA[1])
21.     return Dientich(a,b,c)
22. pA = eval(input("Nhập tọa độ x, y của điểm A: "))
23. pB = eval(input("Nhập tọa độ x, y của điểm B: "))
24. pC = eval(input("Nhập tọa độ x, y của điểm C: "))
25. S = tinhS(pA,pB,pC)
26. print("Diện tích tam giác ABC:",S)
```

8.8. Cho trước tọa độ 4 đỉnh A, B, C, D, viết chương trình kiểm tra cạnh AB có cắt cạnh CD không?

Thuật toán kiểm tra hai đoạn thẳng có giao nhau hay không như sau:

Hai đoạn thẳng AB và CD cắt nhau khi và chỉ khi đường thẳng ab qua AB cắt đoạn CD và đường thẳng cd qua CD sẽ cắt AB. Giả sử f(x) là phương trình đường thẳng d thì d cắt đoạn AB khi và chỉ khi f(A).f(B) < 0.

Chương trình hoàn chỉnh như sau:

```
1.  # 8.8 Kiểm tra xem đoạn AB có giao với đoạn CD không?
2.  A = eval(input("Nhập tọa độ x, y điểm A: "))
3.  B = eval(input("Nhập tọa độ x, y điểm B: "))
4.  C = eval(input("Nhập tọa độ x, y điểm C: "))
5.  D = eval(input("Nhập tọa độ x, y điểm D: "))
6.  def ptDT(P1,P2):
7.      x1,y1 = P1
8.      x2,y2 = P2
9.      a = y2 - y1
10.     b = x1 - x2
11.     c = x2*y1 - x1*y2
12.     return a,b,c
13. def callfunc(a,b,c,P):
14.     x,y = P
15.     return a*x + b*y + c
16. def Cat(A,B,C,D):
17.     a1,b1,c1 = ptDT(A,B)
18.     a2,b2,c2 = ptDT(C,D)
19.     if callfunc(a1,b1,c1,C)*callfunc(a1,b1,c1,D) < 0 \
20.        and callfunc(a2,b2,c2,A)*callfunc(a2,b2,c2,B) < 0:
21.         return True
22.     else:
23.         return False
24.
25. if Cat(A,B,C,D):
26.     print("AB và CD giao nhau.")
27. else:
28.     print("AB và CD không giao nhau.")
```

8.9. Thiết lập một module **MyMatrix** bao gồm các phương thức bổ sung cho đối tượng List. Thiết lập các hàm sau:

- **isMatrix(A)**, kiểm tra xem list A có phải là biểu diễn ma trận hay không. Trả lại True nếu đúng và False nếu sai.

- **ChangeRow(A, i, j)**, hàm thay đổi 2 hàng i, j của Ma trận A. Nếu A không là ma trậ hoặc không thể thực hiện được thì trả lại False, ngược lại nếu thực hiện thành công thì t lại True.

- **ChangeColumn(A, i, j)**, thủ tục thay đổi 2 cột i, j của Ma trận A. Nếu A không là n trận hoặc không thể thực hiện được thì trả lại False, ngược lại nếu thực hiện thành côi thì trả lại True.

- **isSquare(A)**, hàm kiểm tra xem ma trận A có phải là ma trận vuông hay không.

- **GetSymmetry(A)**, thủ tục lấy đối xứng qua đường chéo chính của Ma trận vuông . Hàm trả lại ma trận A^T là đối xứng của A.

```
1.  # 8.9 MyMatrix Module
2.  def isMatrix(A):
3.      res = all(isinstance(E, list) for E in A)
4.      if (len(A) == 0 or res == False):
5.          return False
6.      for i in range(1, len(A)):
7.          if (len(A[i]) != len(A[0])):
8.              return False
9.      return True
10.
11. def ChangeRow(A, i , j):
12.     if isMatrix(A):
13.         A[i], A[j] = A[j], A[i]
14.
15. def ChangeColumn(A, i ,j):
16.     if isMatrix(A):
17.         for col in A:
18.             col[i], col[j] = col[j], col[i]
19.
20. def isSquare(A):
21.     for row in A:
22.         if len(row) != len(A):
23.             return False
24.     return True
25.
26. def GetSymmetry(A):
27.     if not isSquare(A):
28.         return A
29.     n = len(A)
30.     B = [[0 for j in range(n)] for i in range(n)]
31.     for i in range(n):
32.         for j in range(n):
33.             B[i][j] = A[j][i]
34.     return B
```

8.10. Thiết lập module **MyTriange** chứa các hàm và thủ tục liên quan đến tam giác n sau:

- **isTriange(a, b, c)**, hàm kiểm tra xem 3 số a, b, c có thiết lập được tam giác hay khô Trả lại True nếu đúng và False nếu sai.

- **Perimeter(a, b, c)**, hàm tính chu vi của tam giác.

- **Area(a, b, c)**, hàm tính diện tích tam giác.

```
# 8.10 MyTriangle Module
import math
def isTriangle(a,b,c):
    if a + b <= c or a+c <=b or b+c <= a or a <= 0 or b <= 0
or c <= 0:
        return False
    return True
def Perimeter(a,b,c):
    p = (a + b + c)
    return p
def Area(a,b,c):
    p = (a+b+c)/2
    S = math.sqrt(p*(p-a)*(p-b)*(p-c))
    return S
```

8.11. Chúng ta đã biết hằng số e có thể tính bằng công thức:

$$e = \lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n$$

Thiết lập dãy (list) A bao gồm các phần tử sau:

$1, (1 + 1/2)^2, (1 + 1/3)^3, ..., (1 + 1/n)^n, ...$

Dãy này được thiết lập với độ dài N = 1000. Tính giá trị:

$e - (1 + 1/N)^N$.

```
# 8.11
import math
N = int(input('Nhập số N: '))
A = []
for i in range(1, N):
    A.append(math.pow((1+1/i), i))
for x in A:
    print(x)
print("Sai số với e:")
print(math.e - math.pow((1+1/N), N))
```

8.12. Dùng lệnh Python nào để tính được các giá trị phức sau: e^j, e^{-j}.

Lời giải.

```
>>> import cmath
>>> cmath.exp(1j)
(0.5403023058681398+0.8414709848078965j)
>>> cmath.exp(-1j)
(0.5403023058681398-0.8414709848078965j)
>>>
```

8.13. Cho tam giác ABC với các cạnh có độ dài lần lượt là a, b, c. Các đường cao h_a, h_b, h_c tương ứng với các cạnh a, b, c được tính theo công thức:

$h_a = 2S/a$, $h_b = 2S/b$, $h_c = 2S/c$.

S là diện tích tam giác được tính theo công thức:

$$S = \sqrt{p(p-a)(p-b)(p-c)}.$$

Viết chương trình nhập 3 số a, b, c và tính các giá trị h_a, h_b, h_c. Kết quả thể hiện trên m hình.

```
# 8.13
def calculateHeights(a, b, c):
    from math import sqrt
    p = (a + b + c)/2
    S = sqrt(p*(p-a)*(p-b)*(p-c))
    ha = 2*S/a
    hb = 2*S/b
    hc = 2*S/c
    return ha, hb, hc
while True:
    x = (input("Nhập 3 số a, b ,c : ").split())
    a = int(x[0])
    b = int(x[1])
    c = int(x[2])
    if ((a + b <= c) or (a + c <= b) or (b + c <= a)):
        print("Oops, xin hãy nhập lại")
        continue
    break
print(calculateHeights(a, b, c))
```

8.14. Cho trước số tự nhiên N. Sử dụng các hàm của module random để viết thủ tục si một hoán vị ngẫu nhiên của dãy 1, 2, ..., N. Kết quả thể hiện trên màn hình.

Bài tập này là trường hợp riêng của bài 8.15. Xem lời giải chi tiết của bài 8.15.

8.15. Cho trước dãy (list) A bao gồm N phần tử. Sử dụng các hàm của module random viết thủ tục sinh một hoán vị ngẫu nhiên của dãy A. Kết quả thể hiện trên màn hình.

Có nhiều cách thiết lập một hoán vị ngẫu nhiên của một dãy số cho trước. Sau đây một vài cách.

*Cách 1. Sử dụng hàm **shuffle()** từ module random.*

```
def permute_1(A):
    from random import shuffle
    B = A.copy()
    shuffle(B)
    return B
```

*Cách 2. Sử dụng hàm **randint()** từ module random.*

```
def permute_2(A):
    from random import randint
    B = A.copy()
    n = len(B)
    for i in range(n-1):
        j = randint(i+1,n-1)
        B[i],B[j] = B[j],B[i]
    return B
```

*Cách 3. Sử dụng hàm **choice()** và cách làm lấy từ Scratch.*

```
def permute_3(A):
    from random import choice
    B = []
    C = A.copy()
    n = len(C)
```

```
    for i in range(n):
        k = choice(C)
        B.append(k)
        C.remove(k)
    return B
```

8.16. Viết hàm số kiểm tra xem số tự nhiên N có phải là số chính phương hay không. Gợi ý: Có thể sử dụng hàm isqrt() của module math.

Lời giải.

```
def is_square(N):
    from math import isqrt
    return N == math.isqrt(N)**2 #Python 3.8 trở lên mới có isqrt
```

8.17. Viết hàm số kiểm tra xem số tự nhiên N có phải là số nguyên tố hay không, yêu cầu sử dụng ít nhất phép chia lấy số dư. Gợi ý: Dùng hàm floor() của module math.

Lời giải.

```
def isPrime(n):
    from math import sqrt, floor
    for k in range(2,floor(sqrt(n))):
        if n%k == 0:
            return False
    return True
```

8.18. Viết hàm sinh generator Primes(n) trả lại dãy các số nguyên tố không vượt quá số tự nhiên n.

Cách 1. Dùng sàng nguyên tố để lọc danh sách các số nguyên tố.

```
def generate_Primes(n):
    is_prime = [True for i in range(n + 1)]
    is_prime[0] = False
    is_prime[1] = False
    k = 2
    while k*k < n:
        for i in range(k*k, n+1,k):
            is_prime[i] = False
        k = k + 1
    return (i for i in range(n+1) if is_prime[i] == True)
```

Cách 2. Dùng hàm tường minh để xác nhận số nguyên tố.

```
def isPrime(n):
    from math import sqrt, floor
    for k in range(2,floor(sqrt(n))):
        if n%k == 0:
            return False
    return True
def genPrimes(n):
    for k in range(2,n+1):
        if isPrime(k):
            yield k
```

8.19. Viết hàm tính diện tích tam giác biết độ dài 2 cạnh a, b và góc α (tính số đo góc theo độ °), biết công thức tính diện tích là S = absin(α)/2.

Vì hàm sin() trong module math có tham số được tính mặc định theo radian nên chúng ta cần thêm hàm radians() để chuyển đổi từ độ sang radian.

```
def dientich(a,b,alpha):
    from math import sin, radians
    return a*b*sin(radians(alpha))/2
```

8.20. Viết hàm tính chu vi và diện tích hình tròn biết bán kính R. Gợi ý: Sử dụng hằng s Pi của module math.

Lời giải.

```
def chuvi_dientich(R):
    from math import pi
    return 2*pi*R, pi*R*R
```

Chương 9: Xâu ký tự

9.1. Nếu S = "Việt Nam" thì biểu thức sau thể hiện gì?

```
S[ :  : -1]
```

a) "Nam Việt"

b) "teejiV maN"

c) "maN tệiV"

d) Chương trình báo lỗi.

Đáp án đúng: c).

9.2. Đoạn chương trình sau sẽ in ra kết quả gì?

```
msg = "Hà Nội Đà Nẵng Cần Thơ"
for w in msg.split():
  print(w[0], end = " ")
```

a) Hà Nội Đà Nẵng Cần Thơ

b) HàNộiĐàNẵngCầnThơ

c) HNĐNCT

d) H N Đ N C T

Đáp án đúng: d).

9.3. Kết quả các lệnh sau là gì?

```
>>>s = "abcdefg"
>>>s[-1 : 0 : -1]
```

a) "" (xâu rỗng)

b) "gfedcba"

c) "abcdefg"

d) "gfedcb"

Đáp án đúng: d).

9.4. Kết quả các lệnh sau là gì?

```
>>>s = "abcdefg"
>>>s[-1 :  : -1]
```

a) "" (xâu rỗng)

b) "gfedcba"

c) "abcdefg"

d) "gfedcb"

Đáp án đúng: b).

9.5. Cho trước xâu ký tự s. Viết đoạn chương trình tách các ký tự là số của xâu trên và chuyển vào một dãy A.

Ví dụ nếu s = "01-tháng-5-năm 2020" thì kết quả dãy A sẽ là:

A = [0, 1, 5, 2, 0, 2, 0].

```
1.  # 9.5
2.  def Tach_So(s):
3.      so = []
4.      for ch in s:
5.          if "0" <= ch <="9":
6.              so.append(int(ch))
7.      return so
8.  s = "01-tháng-5-năm 2020"
9.  so = Tach_So(s)
10. print(so)
```

9.6. Giả sử s là một xâu ký tự bất kỳ. Khi đó lệnh sau:

```
>>>len(s.split())
```

Có ý nghĩa gì?

Lệnh này có ý nghĩa trả lại số các từ đơn của xâu s.

9.7. Hãy giải thích ý nghĩa đoạn chương trình sau:

```
>>>S = " Bui  Viet   Ha"
>>>S = S.strip()
>>>S1 = S[ : S.find(" ")]
```

Kết quả xâu ký tự S1 thu được là gì?

*Kết quả xâu S1 thu được là phần **họ** của họ tên người được ghi trong xâu S. Trong ví c cụ thể của bài này kết quả thu được là S1 = "Bui".*

9.8. Đoạn chương trình sau cho kết quả là gì?

```
1. S = "Bùi Việt Hà"
2. A = S.split()
3. A = A[ :  : -1]
4. S1 = " ".join(A)
5. S2 = S1[0 : S1.find(" ")]
6. print(S2)
```

*Kết quả thu được là phần **tên** của họ tên người được ghi trong xâu S. Trong ví dụ cụ t của bài này kết quả thu được là S2 = "Hà".*

9.9. Lệnh sau cho kết quả như thế nào?

```
>>> ord("ABC")
```

a) 65

b) h41

c) 1000001

d) Chương trình báo lỗi.

Đáp án đúng: d).

9.10. Kết quả của chương trình sau sẽ in như thế nào?

```
1. s = "ab2vc4f7h8"
2. kq = ""
3. for ch in s:
4.     if not (ch >= '0' and ch <='9'):
5.         kq = kq + ch
```

```
6. print(kq)
```

a) ab2vc4f7h8

b) abvcfh

c) 2478

d) Chương trình báo lỗi.

Đáp án đúng: b).

9.11. Viết thủ tục tính số các từ đơn của một đoạn văn bản - xâu ký tự bất kỳ. Ví dụ với xâu: s = " Hà Nội thủ đô của Việt Nam ", thì kết quả trả lại là 7.

Lời giải.

```
def wlength(s):
    return len(s.split())
```

9.12. Viết thủ tục **fullstrip(s)** có chức năng chỉnh sửa dấu cách của xâu ký tự bất kỳ s, xóa dấu cách đầu, cuối và xóa các dấu cách thừa bên trong xâu s.

Ví dụ với s = " Hà Nội thủ đô của Việt Nam " thì kết quả phải là:

fullstrip(s) = "Hà Nội thủ đô của Việt Nam".

Lời giải.

```
def fullstrip(s):
    s = " ".join(s.split())
    return s
```

9.13. Viết hàm số **TachTen(Hoten)** có chức năng tách phần tên tiếng Việt của một họ tên đầy đủ Hoten. Ví du Hoten = "Nguyễn Quang Bình" thì hàm sẽ trả lại "Bình".

Lời giải.

```
def TachTen(Hoten):
    res = Hoten.strip().split(" ")
    final = res[len(res)-1]
    return final
```

9.14. Cho trước (hoặc nhập từ bàn phím) xâu ký tự Str. Hãy đếm số các ký tự là số trong xâu trên.

Lời giải.

Cách 1: Sử dụng tính chất so sánh của ký tự với chữ số.

```
1. s = "1 con vịt, 22 con sư tử, 32 con tê giác"
2. print(s)
3. number = 0
4. for c in s:
5.     if "0" <= c <= "9":
6.         number = number + 1
7. print(number)
```

*Cách 2: Sử dụng phương thức **isdigit()** của xâu ký tự.*

```
1. s = "1 con vịt, 22 con sư tử, 32 con tê giác"
2. print(s)
3. numbers = sum(c.isdigit() for c in s)
4. print(numbers)
```

9.15. Giải thích ý nghĩa của chương trình sau:

```
1. sin = "Chiến khu ATK Việt Bắc"
2. sout = ""
3. for ch in sin:
4.     sout = sout + chr(ord(ch) + 2)
5. print(sout)
```

Kết quả chương trình in ra là: **Ejkếp"mjw"CVM"Xkiv"Dắc**

*Đây là chương trình mã hóa một đoạn văn bản được cho bởi xâu vào **sin**. Kết quả mã hóa cho trong xâu ra là **sout**. Việc mã hóa như sau: với mỗi ký tự của sin sẽ mã hóa bằng ký tự với số thứ tự trong bảng mã Unicode cộng thêm 2. Việc mã hóa này sẽ dễ dàng được giải mã.*

9.16. Giải thích ý nghĩa của chương trình sau.

```
1. sin = '''Ejkếp"mjw"CVM"Xkiv"Dắc'''
2. sout = ""
3. for ch in sin:
4.     sout = sout + chr(ord(ch) - 2)
5. print(sout)
```

Kết quả đầu ra của chương trình trên là gì? So sánh với chương trình bài 15.

*Đây là chương trình giải mã cho chương trình mã hóa đã cho trong bài 9.15. Kết quả đầu ra của chương trình này là **Chiến khu ATK Việt Bắc**.*

9.17. Hãy giải thích kết quả chương trình sau:

```
1. sin = "Chiến khu ATK Việt Bắc"
2. sout = ""
3. for ch in sin:
4.     if ch == " ":
5.         sout = sout + " "
6.     else:
7.         sout = sout + chr(ord(ch) + 2)
8. print(sout)
```

*Chương trình này thực hiện việc mã hóa xâu đầu vào sin tương tự bài 9.15. Điểm khác biệt là chỉ mã hóa từng từ đơn mà không mã hóa dấu cách. Kết quả xâu được mã hóa sẽ là: **Ejkếp mjw CVM Xkiv Dắc**.*

9.18. Viết hàm số Component(S,k) có ý nghĩa như sau:

S: Xâu ký tự đầu vào.

k: Số tự nhiên.

Kết quả hàm số trả lại thành phần, từ đơn thứ k của xâu S. Nếu số k nằm ngoài phạm vi cho phép thì hàm trả lại xâu rỗng.

Ví dụ:

S = "Hà Nội là thủ đô của Việt Nam".

Component(S, 2) = "Nội".

Component(S, 8) = "Nam".

Component(S, 9) = "".

Lời giải.

```
def Component(S,k):
    A = S.split()
    if (k > len(A) or k < 0):
        return ""
    else:
        return A[k-1]
```

9.19. Cho trước (hoặc nhập từ bàn phím) xâu ký tự Str. Hãy đếm số các ký tự là chữ cái tiếng Anh (kể cả chữ hoa, chữ thường) trong xâu Str.

Lời giải.

Cách 1: Sử dụng so sánh từng ký tự của xâu s với các ký tự chữ cái tiếng Anh.

```
1. # 9.19
2. s = "asdfkasdflasDfl222@424huy*&%$%&*("
3. letters = 0
4. for c in s:
5.     if "a" <= c <= "z" or "A" <= c <= "Z":
6.         letters = letters + 1
7. print(s)
8. print(letters)
```

*Cách 2: Sử dụng phương thức **isalpha()** của xâu ký tự.*

```
1. # 9.19
2. s = "asdfkasdflasDfl222@424huy*&%$%&*("
3. letters = sum(c.isalpha() for c in s)
4. print(s)
5. print(letters)
```

9.20. Cho trước (hoặc nhập từ bàn phím) xâu ký tự Str, đếm số các ký tự không là số, không là chữ cái tiếng Anh trong xâu Str.

Tương tự các bài trên, chúng ta đưa ra 3 lời giải của bài tập này.

Cách 1:

```
1. # 9.20
2. s = "asdfkasdflasDfl222@424huy*&%$%&*("
3. number = 0
4. for c in s:
5.     if not "0"<=c<="9" and not "a"<=c<="z" and not "A"<=c<="Z":
6.         number = number + 1
7. print(number)
```

Cách 2:

```
1. # 9.20
2. s = "asdfkasdflasDfl222@424huy*&%$%&*("
3. number = 0
4. for c in s:
5.     if not c.isdigit() and not c.isalpha():
6.         number = number + 1
7. print(number)
```

Cách 3:

```
1. # 9.20
2. s = "asdfkasdflasDfl222@424huy*&%$%&*("
3. number = sum(not c.isdigit() and not c.isalpha() for c in s)
4. print(number)
```

9.21. Cho trước 2 xâu Str1, Str2. Viết chương trình:

a) Kiểm tra xem xâu Str2 có nằm trong Str1 hay không?

b) Nếu có thì Str2 được xuất hiện (có thể chồng lên nhau) trong Str1 bao nhiêu lần.

c) Tương tự b) nhưng chỉ tính số lần xuất hiện không chồng lên nhau.

Lời giải.

- Hàm kiểm tra xem xâu s2 có nằm trong xâu s1 không?

```
def findin(s1, s2):
    return s1.find(s2) != -1
```

- Hàm tính xem xâu s2 xuất hiện không chồng lên nhau trong s1 bao nhiêu lần.

```
def findStringWithoutOverlapping(s1,s2):
    count = 0
    if (len(s2) > len(s1)):
        return 0
    while (s1.find(s2) != -1):
        s1 = s1.replace(s2,"",1)
        count += 1
    return count
```

- Hàm tính xem xâu s2 xuất hiện (cho phép chồng lên nhau) trong s1 bao nhiêu lần.

```
def findStringWithOverlapping(s1,s2):
    count = 0
    if (len(s2) > len(s1)):
        return 0
    while (s1.find(s2) != -1):
        s1 = s1.replace(s2[0],"",1)
        count += 1
    return count
```

Chương trình hoàn chỉnh thực hiện cả 3 yêu cầu trên có thể như sau:

```
1.  # 9.21.
2.  def findin(s1, s2):
3.      return s1.find(s2) != -1
4.
5.  def findStringWithoutOverlapping(s1,s2):
6.      count = 0
7.      if len(s2) > len(s1):
8.          return 0
9.      while s1.find(s2) != -1:
10.         s1 = s1.replace(s2,"",1)
11.         count += 1
12.     return count
13.
14. def findStringWithOverlapping(s1,s2):
15.     count = 0
16.     if len(s2) > len(s1):
17.         return 0
18.     while s1.find(s2) != -1:
19.         s1 = s1.replace(s2[0],"",1)
20.         count += 1
21.     return count
```

```
22.
23. s1 = "abababababaab"
24. s2 = "aba"
25. print(s2,"xuất hiện trong",s1,"?")
26. print(findin(s1,s2))
27. print("Xuất hiện tách rời:",
    findStringWithoutOverlapping(s1,s2), "lần")
28. print("Có thể chồng lên:", findStringWithOverlapping(s1,s2),
    "lần")
```

Kết quả chạy chương trình trên như sau:

```
aba xuất hiện trong abababababaab ?
True
Xuất hiện tách rời: 3 lần
Có thể chồng lên: 5 lần
```

9.22. Cho trước (hoặc nhập từ bàn phím) xâu ký tự Str, in ra xâu thu được từ Str bằng cách viết theo thứ tự ngược lại.

```
1. # 9.22.
2. s = input("Nhập vào 1 xâu: ")
3. u = s[ : : -1]
4. print(u)
```

9.23. Cho trước 2 xâu Str1, Str2, viết chương trình trộn 2 xâu trên lại, in ra kết quả. Việc trộn xâu theo nguyên tắc sau:

- Lần lượt từ trái sang phải, viết các ký tự của Str1, sau đó đến Str2.

- Nếu 1 trong 2 xâu kết thúc thì chỉ viết ra các ký tự của xâu còn lại.

```
1. # 9.23
2. s1 = input("Nhập xâu 1: ")
3. s2 = input("Nhập xâu 2: ")
4. n1 = len(s1)
5. n2 = len(s2)
6. n = min(n1,n2)
7. s = ""
8. for i in range(n):
9.     s = s + s1[i] + s2[i]
10. if n1 > n2:
11.     for i in range(n,n1):
12.         s = s + s1[i]
13. else:
14.     for i in range(n,n2):
15.         s = s + s2[i]
16. print("s1 =",s1)
17. print("s2 =",s2)
18. print(s)
```

9.24. Cho trước (hoặc nhập từ bàn phím) xâu ký tự Str. Hãy xóa đi tất cả các ký tự của Str không là số, xâu còn lại sẽ là số. Hãy in ra xâu kết quả này và thông báo nó là nguyên tố hay không.

```
1. # 9.24
2. def isPrimary(n):
3.     if n < 2:
4.         return False
```

```
    else:
        k = 2
        while k < n and n%k != 0:
            k = k + 1
        if k < n:
            return False
        else:
            return True
s = input("Nhập xâu bất kỳ: ")
snum = ""
for ch in s:
    if "0" <= ch <= "9":
        snum = snum + ch
if snum != "":
    n = int(snum)
    print("Số còn lại là",n)
    if isPrimary(n):
        print("Số này là nguyên tố.")
    else:
        print("Số này không là nguyên tố.")
```

9.25. Cho trước (hoặc nhập từ bàn phím) xâu ký tự Str. Hãy xóa đi tất cả các ký tự không là chữ cái tiếng Anh của xâu này. In ra xâu kết quả.

```
# 9.25
s = "Special $#! abc%^&*( characters   spaces 888323"
target = ""
for ch in s:
    if ch.isalpha() or ch == " ":
        target = target + ch
print(s)
print(target)
```

9.26. Cho trước xâu ký tự nhị phân bao gồm toàn các ký tự 0 và 1. Viết chương trình chuyển đổi số nhị phân này sang số thập phân.

```
# 9.26
s = '11111111'
def bitodec(s):
    res = 0
    for i in range(len(s)):
        res = res * 2
        res = res + int(s[i])
    return res
print("Kết quả: " ,bitodec(s))
```

9.27. Cho trước xâu Str. Kiểm tra xem Str có phải là xâu nhị phân (toàn 0, 1) hay không. Nếu có thì chuyển xâu này sang số thập phân. In kết quả số thập phân trên màn hình.

```
# 9.27
s = "11111111"
check = False
c = "01"
for char in s:
    if not char in c:
        check = False
        print("Không phải là nhị phân")
```

```
            break
        else:
            check = True
if check:
    res = 0
    for i in range(len(s)):
        res = res * 2
        res = res + int(s[i])
    print("Kết quả: " ,res)
```

9.28. Cho trước số tự nhiên n được viết dưới dạng hệ thập phân. Viết chương trình chuyển số n từ hệ đếm thập phân sang nhị phân. Kết quả là xâu nhị phân.

```
# 9.28
def dectobi(n):
    s = ""
    while n > 0:
        b = n%2
        n = (n - b)//2
        s = s + str(b)
    return s[ : : -1]
print(dectobi(5))
```

9.29. Cho trước xâu Str. Kiểm tra xem Str có phải là xâu được viết trong hệ hex (0, 1, ..., 9, A, B, C, D, E, F) hay không. Nếu có thì chuyển xâu này sang số thập phân. In kết quả số thập phân trên màn hình.

Để làm bài tập này chúng ta cần thiết lập 2 hàm số:

*Hàm **isHex(s)** kiểm tra xem xâu s có phải được viết trong hệ hexa hay không.*

*Hàm **hextodec(s)** sẽ chuyển đổi số từ dạng hex (xâu hex) sang số thập phân.*

```
# 9.29
def isHex(s):
    check = True
    hexdigits = "0123456789ABCDEF"
    s = s.upper()
    for char in s:
        if char not in hexdigits:
            check = False
            break
    return check
def hextodec(s):
    if not isHex(s):
        return -1
    sample = "0123456789ABCDEF"
    dec = 0
    for index in range(len(s)):
        d = s[index]
        value = sample.index(d)
        p = len(s) -(index+1)
        dec = dec + value*(16**p)
    return dec
s = input("Nhập xâu hex: ")
dec = hextodec(s)
print(dec)
```

9.30. Cho trước (hoặc nhập từ bàn phím) xâu ký tự Str. Hãy dồn các số là chữ số sang bên trái (giữ đúng thứ tự) và in ra kết quả. Ví dụ:

Input: abc9k00ty8n0.

Output: 90080abcktyn.

Lời giải.

Cách 1:

```
# 9.30
sin = input("Nhập 1 xâu: ")
s1 = ''.join([c for c in sin if c.isdigit()])
s2 = ''.join([c for c in sin if not c.isdigit()])
sout = s1 + s2
print(sout)
```

Cách 2:

```
# 9.30
sin = input("Nhập 1 xâu: ")
s1 = ""
s2 = ""
for ch in sin:
    if "0" <= ch <= "9":
        s1 = s1 + ch
    else:
        s2 = s2 + ch
sout = s1 + s2
print(sout)
```

9.31. Cho trước 2 xâu Str và St, số k < length(Str). Viết chương trình chèn xâu St vào Str tại vị trí ký tự thứ k. In kết quả ra màn hình.

```
# 9.31
Str = "Bùi Vũ Huy"
St = "Hoàng "
print(Str)
while True:
    k = int(input("Nhập vị trí bạn muốn chèn: "))
    if 0 > k:
        print("Bạn không thể chèn vào vị trí đó")
        continue
    break
Str = Str[:k] + St + Str[k:]
print(Str)
```

9.32. Nhập từ bàn phím một họ tên người Việt Nam đầy đủ, ví dụ: Bùi Việt Hà. Viết chương trình tách và in ra họ, đệm, tên của người này. Ví dụ:

Input: Bùi Việt Hà. Output: họ - Bùi, đệm - Việt, tên - Hà.

Input: Nguyễn Khang. Output: họ - Nguyễn, đệm không có, tên - Khang.

Input: Bạch Thị Thúy Hoa. Output: họ - Bạch, đệm - Thị Thúy, tên - Hoa.

```
# 9.32
Hoten = input("Nhập họ tên đầy đủ: ")
res = Hoten.strip().split()
dem = " ".join(res[1:len(res)-1])
```

```
print("họ - :", res[0])
if not dem:
    print("đệm không có ")
else:
    print("đệm -: ",Dem)
print("tên -: ", res[len(res)-1])
```

9.33. Nhập một họ tên đầy đủ, viết chương trình sửa, xóa đi các dấu cách thừa của họ tên này. Chú ý dấu cách thừa có thể ở giữa các họ, đệm, tên hoặc ở đầu và cuối xâu.

```
# 9.33.
Hoten = input("Nhập họ tên đầy đủ: ")
Hoten = " ".join(Hoten.split())
print(Hoten)
```

9.34. Cho trước xâu Str là một đoạn văn bản hoàn chỉnh (có thể nhiều dòng). Nhập một từ đơn, hãy tìm xem trong văn bản Str chứa bao nhiêu từ đơn này.

```
# 9.34
Str = input("Nhập một văn bản hoàn chỉnh: ")
res = Str.strip().split(" ")
word = input("Nhập từ cần đếm: ")
print(res.count(word))
```

9.35. Nhập một xâu bất kỳ từ bàn phím. Viết chương trình kiểm tra nếu thấy có 2 ký tự liền nhau và giống nhau thì xóa đi một ký tự. Ví dụ nhập vào là aabbbccc thì đưa ra kết quả abc.

```
# 9.35
string = input("Nhập 1 xâu ký tự: ")
s = ""
prev = ""
for ch in string:
    if ch != prev:
        s = s + ch
        prev = ch
print(s)
```

9.36. Viết chương trình nhập lần lượt thông tin học sinh trong lớp theo các thông số sau:

- Nhập họ tên học sinh.

- Nhập điểm số môn Toán (số thập phân với 1 chữ số sau dấu phẩy).

- Nhập điểm số môn Văn (số thập phân với 1 chữ số sau dấu phẩy).

Việc nhập dữ liệu kết thúc khi người dùng nhập một xâu rỗng.

Sau đó in ra màn hình họ tên học sinh, điểm số môn Toán, Văn theo mẫu sau:

Họ tên	Toán	Văn
Nguyễn Văn Quang	8.5	9.1
Bùi Quang Bình	7.0	8.6
Trần Thu Hà	10	9.8
Lê Quang My	9.7	5.6

Yêu cầu thể hiện ra như sau:

- Họ tên học sinh được căn trái, chiếm khoảng rộng 30 ký tự.

- Điểm Toán, chiếm khoảng rộng 5, căn phải.

- Điểm Văn, chiếm khoảng rộng 5, căn phải.

```
# 9.36
HS = []
while True:
    hoten = input("Nhập họ tên học sinh: ")
    if not hoten:
        break
    dToan = float(input("Nhập điểm TB Toán: "))
    dVan = float(input("Nhập điểm TB Văn: "))
    HS.append([hoten,dToan,dVan])
print(HS)
print("Họ tên học sinh                    Toán  Văn")
for hoten,dToan,dVan in HS:
    print("{:<30} {:5.1f} {:5.1f}".format(hoten,dToan,dVan))
```

9.37. Xâu ký tự có độ dài n được gọi là đối xứng nếu s[k] = s[n-k-1] với k = 0, 1, 2, .., n-1. Viết hàm số **isSymmetry(s)** kiểm tra xem xâu S có phải là đối xứng hay không.

Lời giải.

```
def isSymmetry(s):
    for k in range(len(s)//2):
        if s[k] != s[len(s)-k-1]:
            return False
        return True
```

9.38. Viết chương trình nhập từ bàn phím một danh sách tên học sinh trong lớp. Cần nhập họ đệm và tên đầy đủ. Ví dụ có thể nhập một danh sách như sau:

```
Nguyễn Văn Quang
Bùi Quang Thành
Trần Văn An
Lê Quang Bình
```

Chương trình sẽ sắp xếp lại danh sách học sinh theo thứ tự từ điển nhưng theo nguyên tắc sắp xếp tên trước họ đệm sau. Với bộ dữ liệu trên, kết quả sắp xếp sẽ phải như sau:

```
Trần Văn An
Lê Quang Bình
Nguyễn Văn Quang
Bùi Quang Thành
```

Lời giải.

```
# 9.38
def TachTen(hoten):
    hoten_list = hoten.split()
    ten = hoten_list[len(hoten_list)-1]
    return ten
hs = []
while True:
    hoten = input("Nhập họ tên học sinh: ")
    if not hoten:
        break
    hs.append(hoten)
hsgoc = hs.copy()
hsfull = []
```

```
for s in hs:
    hsfull.append(TachTen(s)+s)
n = len(hs)
for i in range(n-1):
    for j in range(i,n):
        if hsfull[i] > hsfull[j]:
            hsfull[i],hsfull[j] = hsfull[j],hsfull[i]
            hs[i],hs[j] = hs[j],hs[i]
print("Danh sách học sinh gốc:")
for s in hsgoc:
    print(s)
print("Danh sách học sinh sau khi sắp xếp:")
for s in hs:
    print(s)
```

9.39. Cho trước (hoặc nhập từ bàn phím) xâu ký tự Str. Hãy tìm một xâu con có độ dài cực đại và bao gồm các phần tử giống nhau của xâu trên. Ví dụ nếu xâu vào là aabbbcc thì output là bbb.

Lời giải sau chỉ yêu cầu một vòng lặp for thực hiện n chạy dọc theo xâu Str. Bộ dữ liệu xâu con được xác định bởi các tham số:

- Độ dài xâu: len.

- Vị trí đầu của xâu: ic.

Khi di chuyển dọc theo xâu Str thì bộ dữ liệu (len, ic luôn được cập nhật và so sánh với bộ thông tin của xâu con cực đại là (lenmax, imax).

```
# 9.39
s = input("Nhập xâu bất kỳ: ")
n = len(s)
lenmax = 1 # độ dài xâu con cực đại
imax = 0   # vị trí đầu xâu con cực đại
len = 1    # độ dài hiện thời
ic = 0     # vị trí đầu xâu hiện thời
for i in range(1, n):
    if s[i] == s[i-1]:
        len = len + 1
    else:
        if len > lenmax:
            lenmax = len
            imax = ic
        len = 1
        ic = i
if len > lenmax:
    lenmax = len
    imax = ic
print("Độ dài cực đại:",lenmax)
print("Xâu con cực đại:",s[imax:imax+lenmax])
```

9.40. Cho trước (hoặc nhập từ bàn phím) 2 xâu ký tự Str1 và Str2. Hãy tìm ra xâu con chung có độ dài lớn nhất của 2 xâu trên.

Ý tưởng thuật toán của bài tập này tương tự như bài 9.39. Chúng ta sẽ chỉ cần một vòng duyệt dọc theo xâu s1 và kiểm tra thông tin của xâu con chung hiện thời.

Các tham số strc, ic, len mô tả xâu con chung hiện thời, vị trí đầu của xâu và độ dài xâu.

Các tham số ztrmax, imax, lenmax mô tả xâu con chung cực đại của 2 xâu s1, s2.

Biến nhớ status có ý nghĩa:

= True, nếu ký tự hiện thời nằm trong cả 2 xâu s1, s2.

= False, nếu ký tự hiện thời không là chung trong 2 xâu s1, s2.

```
# 9.40
s1 = "ab123bcf4567mm01234n"
s2 = "0123456789"
n = len(s1)
lenmax = 0 # độ dài xâu con cực đại
imax = -1  # vị trí đầu xâu con cực đại
strmax = ""# xâu cực đại
len = 0    # độ dài hiện thời
ic = -1    # vị trí đầu xâu hiện thời
strc = ""  # xâu hiện thời
status = False
for i in range(n):
    ch = s1[i]
    if strc + ch in s2:
        if status:
            len = len + 1
            strc = strc + ch
        else:
            len = 1
            ic = i
            strc = ch
            status = True
    else:
        if len > lenmax:
            lenmax = len
            strmax = strc
            imax = ic
        if ch in s2:
            len = 1
            ic = i
            strc = ch
            status = True
        else:
            len = 0
            ic = -1
            strc = ""
            status = False
if len > lenmax:
    lenmax = len
    strmax = strc
    imax = ic
print("Xâu con chung cực đại là:",strmax)
```

Chương 10: Đọc và ghi tệp

Các bài tập sau sẽ cùng sử dụng tệp "DL.txt" có dạng:

```
This is text file
The second line
The last
```

10.1. Giả sử chúng ta viết đoạn chương trình sau:

```
file = open("DL.txt", "r")
text = file.read()
```

Hỏi biến nhớ **text** sẽ thuộc kiểu dữ liệu nào?

a) str

b) list

c) tuple

d) file object

Đáp án đúng: a)

10.2. Giả sử chúng ta viết đoạn chương trình sau:

```
file = open("DL.txt", "r")
text = file.readlines()
```

Hỏi biến nhớ **text** sẽ thuộc kiểu dữ liệu nào?

a) str

b) list

c) tuple

d) file object.

Đáp án đúng: b)

10.3. Chương trình sau in ra kết quả như thế nào?

```
File = open("DL.txt","r")
for line in File:
    print(line)
File.close()
```

a)

"This is text file"

"The second line"

"The last"

b)

This is text file

The second line

The last

c)

"This is text file\n The second line\n The last\n"

Đáp án đúng: b).

10.4. Viết chương trình nhập dữ liệu từ tệp text file có dạng như sau:

- Dòng đầu tiên là số tự nhiên n.

- Dòng thứ 2 là n số được viết cách nhau bởi dấu cách.

Ví dụ một tệp như vậy.

```
5
19 -2 9.8 7 -10
```

Chương trình cần đọc dữ liệu đưa vào biến nhớ n và list A, sau đó in n và A ra màn hình.

```
1.  # 10.4
2.  fdata = "Data.dat"
3.  A = []
4.  f = open(fdata,"r")
5.  n = int(f.readline())
6.  line = f.readline()
7.  line = line.split()
8.  for e in line:
9.      A.append(float(e))
10. f.close()
11. print(n)
12. print(A)
```

10.5. Viết chương trình thực hiện công việc ngược lại so với bài tập trên.

Giả sử cho trước số tự nhiên n và dãy số A bao gồm n số hạng bất kỳ.

Viết chương trình ghi dữ liệu ra một text file có dạng sau:

- Dòng đầu tiên là số tự nhiên n.

- Dòng thứ 2 là các số của dãy A được ghi cách nhau bởi dấu cách.

Ví dụ ghi ra tệp như sau:

```
5
19 -2 9.8 7 -10
```

Lời giải.

```
1.  # 10.5
2.  A = [19.0, -2.0, 9.8, 7.0, -10.0]
3.  fdata = "Data.dat"
4.  f = open(fdata,"w")
5.  print(len(A),file = f)
6.  for a in A:
7.      print(a,file = f,end = " ")
8.  f.close()
```

10.6. Viết chương trình trên Python thực hiện công việc sau:

- Đọc dữ liệu từ tệp dạng plain text **fin.dat** và đưa dữ liệu ra tệp văn bản **fout.dat**.

- Tệp **fin.dat** có cấu trúc như sau:

    + Dòng đầu tiên là số tự nhiên n.

+ n dòng tiếp theo, mỗi dòng là một dãy các số nguyên hoặc thực, cách nhau bởi dấu cách. Số lượng các phần tử của mỗi dãy có thể không bằng nhau.

- Tệp **fout.dat** có cấu trúc như sau:

+ Dòng đầu tiên ghi số S là tổng của tất cả các số trong các dãy số từ tệp **fin.dat**.

+ n dòng tiếp theo, dòng thứ i ghi số S_i là tổng các số của dãy số tương ứng từ tệp **fin.dat**.

Ví dụ:

fin.dat
3
-1 10 5 4 10
15 -11 80 -10
3 4 7 -8

fout.dat
108
28
74
6

Lời giải.

```
1.  # 10.6
2.  fin = "fin.dat"
3.  fout = "fout.dat"
4.  fi = open(fin,"r")
5.  n = int(fi.readline())
6.  S = []
7.  for line in fi:
8.      line = line.split()
9.      A = []
10.     for x in line:
11.         A.append(int(x))
12.     S.append(sum(A))
13. fi.close()
14. fo = open(fout,"w")
15. print(sum(S),file=fo)
16. for k in range(n):
17.     print(S[k],file = fo)
18. fo.close()
```

10.7. Cho trước dãy (list) A bao gồm các số bất kỳ. Viết chương trình ghi ra tệp văn bản "DL.dat" có dạng là nhiều dòng, các dòng là các số lấy từ A.

Ví dụ nếu dãy A có dạng:

A = [0, 0.3, 5.7, 19, -1.3, 15.56, 7.0]

thì tệp "DL.dat" sẽ có dạng:

```
0
0.3
5.7
19
-1.3
15.56
7.0
```

```
1.  # 10.7
2.  A = [0, 0.3, 5.7, 19, -1.3, 15.56, 7.0]
3.  fdata = "Data.dat"
4.  f = open(fdata,"w")
5.  for x in A:
6.      print(x,file = f)
7.  f.close()
```

10.8. Bài tập này là ngược lại bài tập 7. Cho trước tệp văn bản "DL.dat" có dạng n dòng, mỗi dòng là một số. Yêu cầu đọc dữ liệu từ tệp này và đưa vào dãy (list) A bao gồm n phần tử vừa đọc từ tệp này.

```
1.  # 10.8
2.  A = []
3.  fdata = "Data.dat"
4.  f = open(fdata,"r")
5.  for line in f:
6.      A.append(float(line))
7.  f.close()
8.  print(A)
```

10.9. Cho trước ma trận A dạng m × n (m hàng, n cột) được mô tả bởi danh sách như sau:

$A = [[a_{11}, a_{12}, \ldots, a_{1n}], [a_{21}, a_{22}, \ldots, a_{2n}], \ldots, [a_{m1}, a_{m2}, \ldots, a_{mn}]]$

Viết chương trình ghi dữ liệu ma trận A ra tệp văn bản dưới dạng sau:

- Dòng đầu tiên ghi 2 số m, n cách nhau bởi dấu cách.

- m hàng tiếp theo, mỗi hàng ghi dãy các số của hàng tương ứng của ma trận A, các số cách nhau bởi dấu cách.

Tệp dữ liệu phải có dạng như sau:

```
m n
a11  a12  ...  a1n
a21  a22  ...  a2n
........
am1  am2  ...  amn
```

Lời giải.

```
1.  # 10.9
2.  A = [[1,2,3],[0,1,0],[9,10,-1],[-3,6,8]]
3.  fdata = "Data.dat"
4.  f = open(fdata,"w")
5.  m = len(A)
6.  n = len(A[0])
7.  print(m,n,file = f)
8.  for i in range(m):
9.      for j in range(n):
10.         print(A[i][j], file = f, end = " ")
11.     print(file = f)
12. f.close()
```

10.10. Bài tập này ngược lại với bài tập 9. Giả sử cho tệp văn bản có dạng:

```
m n
a11  a12  ...  a1n
```

a_{21} a_{22} ... a_{2n}

.....

a_{m1} a_{m2} ... a_{mn}

- Dòng đầu tiên ghi 2 số tự nhiên m, n cách nhau bởi dấu cách.

- m dòng tiếp theo, mỗi dòng có n số, các số cách nhau bởi dấu cách.

Cần viết chương trình đọc dữ liệu từ tệp văn bản trên và đưa vào dãy (list) A có dạng:

A = [[a_{11}, a_{12}, ..., a_{1n}], [a_{21}, a_{22}, ..., a_{2n}], ..., [a_{m1}, a_{m2}, ..., a_{mn}]]

```
# 10.10
A = []
fdata = "Data.dat"
f = open(fdata,"r")
line = f.readline()
line = line.split()
m,n = int(line[0]),int(line[1])
for line in f:
    temp = []
    line = line.split()
    for x in line:
        temp.append(int(x))
    A.append(temp)
f.close()
for i in range(m):
    for j in range(n):
        print(A[i][j],end = " ")
    print()
```

10.11. Viết chương trình đọc dữ liệu từ tệp Input.dat bao gồm n số tự nhiên, mỗi số trên một dòng. Đầu ra là tệp văn bản Output.dat có dạng n dòng, mỗi dòng sẽ ghi số 0 hoặc 1 nếu số tương ứng của tệp Input.dat là hợp số hay nguyên tố.

```
# 10.11
finput = "Input.dat"
foutput = "Output.dat"
def nguyento(n):
    k = 2
    while n%k != 0 and k < n:
        k = k + 1
    if k == n:
        return True
    else:
        return False
fi = open(finput,"r")
fo = open(foutput,"w")
for num in fi:
    num = int(num)
    if nguyento(num):
        print(1,file = fo)
    else:
        print(0,file = fo)
fi.close()
fo.close()
```

20.12. Viết chương trình đọc dữ liệu từ tệp Inp.dat có khuôn dạng bao gồm n số tự nhiên được viết trên một dòng, cách nhau bởi dấu cách. Đầu ra là tệp Out.dat bao gồm n số trên một dòng nhưng các số này đã được sắp xếp theo thứ tự tăng dần.

```
# 10.12
finput = "Inp.dat"
foutput = "Out.dat"
def Read_data():
    A = []
    f = open(finput,"r")
    line = f.readline()
    line = line.split()
    for x in line:
        A.append(int(x))
    f.close()
    return A
def Out_data():
    f = open(foutput,"w")
    B = sortlist(A)
    print(*B,file = f)
    f.close()
def sortlist(A):
    B = A.copy()
    n = len(B)
    for i in range(n-1):
        for j in range(i+1,n):
            if B[i] > B[j]:
                B[i],B[j] = B[j],B[i]
    return B
A = Read_data()
Out_data()
```

10.13. Cho trước dãy số $a_1, a_2, ..., a_n$ được đọc từ tệp Inp.dat có khuôn dạng bao gồm n số tự nhiên được viết trên một dòng, cách nhau bởi dấu cách. Phần tử a_k được gọi là *cực trị* nếu xảy ra một trong hai điều sau: $a_{k-1} < a_k > a_{k+1}$ hoặc $a_{k-1} > a_k < a_{k+1}$. Viết chương trình in ra tệp Out.dat gồm 2 dòng:

- Dòng 1 ghi số lượng các điểm cực trị của dãy trên.

- Dòng 2 ghi ra lần lượt tất cả các phần tử cực trị của dãy trên.

```
# 10.13
finput = "Inp.dat"
foutput = "Out.dat"
def Read_data():
    A = []
    f = open(finput,"r")
    line = f.readline()
    line = line.split()
    for x in line:
        A.append(int(x))
    f.close()
    return A
def process(A):
    B = []
    for k in range(1,len(A)-1):
```

```
            if A[k-1] < A[k] > A[k+1] or A[k-1] > A[k] < A[k+1]:
                B.append(A[k])
    return B
def Out_data():
    f = open(foutput,"w")
    print(len(B),file = f)
    print(*B, file = f)
    f.close()
A = Read_data()
B = process(A)
Out_data()
```

10.14. Viết chương trình đọc tệp dữ liệu Inp.dat bao gồm n dòng, mỗi dòng là một số tự nhiên. Chương trình sẽ đưa kết quả ra tệp Out.dat bao gồm n dòng, mỗi dòng sẽ ghi ra lần lượt các ước số nguyên tố khác nhau của số tương ứng từ tệp Inp.dat. Ví dụ như sau:

Inp.dat
20
25
30

Out.dat
2 5
5
2 3 5

Lời giải.

```
# 10.14
finput = "Inp.dat"
foutput = "Out.dat"
fi = open(finput,"r")
fo = open(foutput,"w")
for line in fi:
    n = int(line)
    k = 2
    while n > 1:
        while n%k != 0:
            k = k + 1
        if n%k == 0:
            print(k,file = fo, end = " ")
            while n%k == 0:
                n = n//k
    print(file = fo)
fi.close()
fo.close()
```

10.15. Tìm hiểu các phương thức của đối tượng kiểu file và viết chương trình sau.

Giả sử có n tệp văn bản với các tên tệp lưu trong một dãy (list). Các tệp này đều có phần mở rộng là *.dat. Mỗi tệp này đều lưu 1 dòng bao gồm dãy các số nguyên (số lượng phần tử trong mỗi dãy có thể khác nhau). Chương trình sẽ lần lượt đọc các dãy số từ từng tệp này, sau đó tự động ghi kết quả là dãy số tương ứng sau khi đã sắp xếp lại theo thứ tự tăng dần. Các tệp đầu ra tương ứng sẽ có tên lấy từ tệp đầu vào sau khi bổ sung thêm "kq" ở đầu tệp.Ví dụ nếu các tệp ban đầu là "Inp1.dat", "Inp2.dat", "Inp3.dat" thì các tệp kết quả sẽ được tự động đặt tên là "kqInp1.dat", "kqInp2.dat", "kqInp3.dat".

Gợi ý: Sử dụng thuộc tính **name** của đối tượng kiểu file.

```
# 10.15
flist = ["Inp1.dat","Inp2.dat","Inp3.dat"]
def sortlist(A):
    B = A.copy()
    n = len(B)
    for i in range(n-1):
        for j in range(i+1,n):
            if B[i] > B[j]:
                B[i],B[j] = B[j],B[i]
    return B
for name in flist:
    fi = open(name,"r")
    fo = open("kq"+name,"w")
    line = fi.readline()
    line = line.split()
    A = []
    for k in range(len(line)):
        A.append(int(line[k]))
    B = sortlist(A)
    print(*B, file = fo)
    fi.close()
    fo.close()
```

Chương 11: Câu lệnh điều kiện

11.1. Giá trị x sau khi thực hiện lệnh sau sẽ là gì?

```
x = 4 ** 2  - 4 * 4 == 0
```

a) True

b) False

c) Không là False và cũng không là True

d) Chương trình báo lỗi.

Đáp án đúng: a).

11.2. Cho đoạn chương trình:

```
>>> a = 5
>>> b = 7
>>> X = a < b or a < 0 and b < 0
```

Giá trị của X là:

a) True

b) False

c) Chương trình báo lỗi.

Đáp án đúng: a).

11.3. Cho đoạn chương trình:

```
>>> a = 5
>>> b = 7
>>> X = (a < b or a < 0) and (b < 0)
```

Giá trị của X là:

a) True

b) False

c) Chương trình báo lỗi.

Đáp án đúng: b).

11.4. Cho đoạn chương trình sau:

```
1. A = [1, 15, -9, 20, -99, 100, 120, 7, 10]
2. B = []
3. for i in A:
4.     if i > 1 and i < 15:
5.         B.append(i)
6. print(B)
```

Dãy B có bao nhiêu phần tử?

a) 1

b) 2

c) 5

d) 9

*Hàm **nguyento(n)** được xác định trước để kiểm tra số tự nhiên n có phải là nguyên tố hay không.*

```
# 11.7
def nguyento(n):
    k = 2
    while n%k != 0 and k < n:
        k = k + 1
    if k == n:
        return True
    else:
        return False
def NextPrime(n):
    k = n + 1
    while not nguyento(k):
        k = k + 1
    return k
```

11.8. Đoạn chương trình sau thực hiện công việc gì?

```
>>> def f(A):
    for i in range(len(A) - 1):
        if A[i] > A[i+1]:
            return False
    return True
>>>A = [2, 1, 3]
>>>f(A)
```

Giá trị của f(A) trong chương trình trên là gì?

a) True

b) False

c) Chương trình báo lỗi.

Đáp án đúng: b)

Hàm f(A) trả lại True nếu A là dãy đơn điệu tăng, ngược lại trả lại False.

11.9. Cho trước dãy (list) A gồm các số bất kỳ.

Viết chương trình tính chỉ số và giá trị phần tử lớn nhất của dãy A.

Cách 1:

```
# 11.9
A = [1,3,5,-1,0,16,100,-8,0,0,1,100,2,3,4,10]
jmax = 0
Max = A[jmax]
for k in range(len(A)):
    if A[k] > Max:
        Max = A[k]
        jmax = k
print(A)
print("Chỉ số max:",jmax,"Giá trị Max:", Max)
```

Cách 2: Sử dụng các hàm của Python.

```
# 11.9b
A = [1,3,5,-1,0,16,100,-8,0,0,1,100,2,3,4,10]
Max = max(A)
```

```
4. jmax = A.index(Max)
5. print(A)
6. print("Chỉ số max:",jmax,"Giá trị Max:", Max)
```

11.10. Cho trước dãy (list) A gồm các số.

Viết chương trình tính chỉ số và giá trị phần tử nhỏ nhất của dãy A.

Lời giải bài tập này hoàn toàn tương tự bài trên, chỉ cần thay đổi max thành min.

Cách 1:

```
1. # 11.10
2. A = [1,3,5,-1,0,16,100,-8,0,0,1,100,2,3,4,10]
3. jmin = 0
4. Mnin = A[jmin]
5. for k in range(len(A)):
6.     if A[k] < Min:
7.         Min = A[k]
8.         jmin = k
9. print(A)
10. print("Chỉ số min:",jmin,"Giá trị Mmin:", Min)
```

Cách 2: Sử dụng các hàm của Python.

```
1. # 11.10b
2. A = [1,3,5,-1,0,16,100,-8,0,0,1,100,2,3,4,10]
3. Min = min(A)
4. jmin = A.index(Min)
5. print(A)
6. print("Chỉ số min:",jmin,"Giá trị Min:", Min)
```

11.11. Nhập số n từ bàn phím và tính tổng $S = 1/1^2 + 1/2^2 + ... + 1/n^2$.

```
1. # 11.11
2. n = int(input("Nhập số tự nhiên: "))
3. s = 0
4. for k in range(1,n+1):
5.     s = s + 1/(k*k)
6. print("Tổng là:",s)
```

11.12. Nhập số n từ bàn phím và tính tổng $S = 1/(1.2) + 1/(2.3) + ... + 1/(n.(n+1))$.

```
1. # 11.12
2. n = int(input("Nhập số tự nhiên: "))
3. s = 0
4. for k in range(1,n+1):
5.     s = s + 1/(k*(k+1))
6. print("Tổng là:",s)
```

11.13. Tính tổng các số của dãy sau:

1 + 5 + 9 + 13 + ... + 205.

Đáp số: 5356.

```
1. # 11.13
2. s = 0
3. k = 1
4. while k <= 205:
5.     s = s + k
6.     k = k + 4
```

```
7. print("Tổng là:",s)
```

11.14. Nhập số n từ bàn phím, in ra các số nguyên tố nhỏ hơn n.

```
1.  # 11.14
2.  n = int(input("Nhập số tự nhiên n: "))
3.  def nguyento(n):
4.      k = 2
5.      while n%k != 0 and k < n:
6.          k = k + 1
7.      if k == n:
8.          return True
9.      else:
10.         return False
11. for k in range(2,n):
12.     if nguyento(k):
13.         print(k, end = " ")
```

11.15. Nhập số n từ bàn phím, in ra khai triển của số n thành tích các thừa số nguyên tố. Ví dụ 20 = 2×2×5.

```
1.  # 11.15
2.  n = int(input("Nhập số tự nhiên n: "))
3.  def KhaitrienNT(n):
4.      print(n,"=",end = " ")
5.      k = 2
6.      first = 1
7.      while n > 1:
8.          while n%k != 0:
9.              k = k + 1
10.         if n%k == 0:
11.             if first == 1:
12.                 print(k,sep = "",end = "")
13.                 first = 0
14.             else:
15.                 print("x",k,sep = "",end = "")
16.             n = n//k
17.             while n%k == 0:
18.                 print("x",k,sep = "",end = "")
19.                 n = n//k
20.         k = k + 1
21. KhaitrienNT(n)
```

11.16. Nhập số n, in ra n số hạng đầu tiên của dãy số Fibonacci.

```
1.  # 11.16. In ra n số Fibonacci đầu tiên
2.  n = int(input("Nhập số tự nhiên n: "))
3.  def Fibonacci(n):
4.      if n == 0:
5.          print(0)
6.      elif n == 1:
7.          print(0,1)
8.      else:
9.          a = 0
10.         b = 1
11.         print(a,b,end = " ")
12.         for k in range(2,n):
```

```
            c = a + b
            print(c, end = " ")
            a = b
            b = c
Fibonacci(n)
```

Ví dụ với n = 30 thì chương trình in ra như sau:

```
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181
6765 10946 17711 28657 46368 75025 121393 196418 317811 514229
```

11.17. Nhập 2 số tự nhiên n, m, tính ước số chung lớn nhất (USCNN) và bội số chung nhỏ nhất (BSCNN) của n, m.

Ta đã biết công thức sau tính USCLN và BSCNN của 2 số m, n:

USCLN(m, n).BSCNN(m, n) = m.n.

```
# 11.17
def USCLN(m,n):
    while m != n:
        if m > n:
            m = m - n
        else:
            n = n - m
    return m
def BSCNN(m,n):
    return m*n//USCLN(m,n)
m,n = eval(input("Nhập 2 số tự nhiên cách nhau bởi dấu phảy m, n: "))
print("USCLN:",USCLN(m,n)," BSCNN:",BSCNN(m,n))
```

11.18. Nhập 3 số tự nhiên n, m, p tính ước số chung lớn nhất của n, m, p.

Để làm bài tập này chúng ta cần chứng minh đẳng thức sau:

USCLN(m, n, p) = USCLN(USCLN(m, n), p)

Để chứng minh công thức trên chúng ta chỉ cần chứng minh mệnh đề sau.

Nếu d là ước số chung của m, n thì d cũng là ước của USCLN(m, n).

Chứng minh

Nhớ lại cách thiết lập USCLN của 2 số tự nhiên m, n như sau: Giả sử u = USCLN(m, n) thì khai triển thành tích các thừa số nguyên tố của u sẽ phải có dạng:

$u = p_1^{k1}p_2^{k2}....p_s^{ks}$ với $p_1, p_2, ..., p_s$ là các thừa số nguyên tố chung trong khai triển của cả m, n, còn $k_1, k_2, ..., k_s$ là số mũ nhỏ nhất của các số mũ tương ứng của $p_1, p_2, ..., p_s$ trong khai triển của m, n.

Nếu d là ước số của cả m và n thì các thừa số nguyên tố trong khai triển của d phải nằm trong các số $p_1, p_2, ..., p_s$ với số mũ tương ứng bắt buộc phải nhỏ hơn hoặc bằng $k_1, k_2, ..., k_s$ tương ứng. Từ đó suy ra d phải là ước số của u. Đó là đpcm.

```
# 11.18
def USCLN(m,n):
    while m != n:
        if m > n:
            m = m - n
        else:
            n = n -m
    return m
```

```
9.  def gcd(m,n,p):
10.     return USCLN(USCLN(m,n),p)
11. m,n,p = eval(input("Nhập 3 số tự nhiên cách nhau bởi dấu phảy
    m, n, p: "))
12. print("USCLN:",gcd(m,n,p))
```

11.19. Nhập 2 số tự nhiên n, m, tính và đếm số các ước chung của n, m.

Vì các ước chung của m, n sẽ đều là ước của USCLN(m, n) nên chỉ cần đếm từ các ước của USCLN này.

```
1.  # 11.19
2.  def USCLN(m,n):
3.      while m != n:
4.          if m > n:
5.              m = m - n
6.          else:
7.              n = n -m
8.      return m
9.  def Counting(m,n):
10.     p = USCLN(m,n)
11.     count = 0
12.     for k in range(1,p + 1):
13.         if p%k == 0:
14.             count = count + 1
15.     return count
16. m,n = eval(input("Nhập 2 số tự nhiên cách nhau bởi dấu phảy m,
    n: "))
17. print("Số các ước số chung là:",Counting(m,n))
```

11.20. Nhập 3 số tự nhiên n, m, p, tính bội số chung nhỏ nhất của n, m, p.

Chúng ta sẽ chứng minh công thức sau, tương tự như với USCLN của 3 số:

BSCNN(m, n, p) = BSCNN(BSCNN(m, n), p)

Chứng minh

Rõ ràng nếu một số là bội đồng thời của BSCNN(m, n) và p thì sẽ đồng thời là bội của m, n.

Ngược lại nếu một số q là bội của m, n, p thì q sẽ là bội của m, n, từ đó suy ra q là bội của BSCNN(m, n). Từ đó suy ra đẳng thức trên.

Từ công thức trên suy ra cách thiết lập BSCNN(m, n, p) như sau.

```
1.  # 11.20
2.  def USCLN(m,n):
3.      while m != n:
4.          if m > n:
5.              m = m - n
6.          else:
7.              n = n -m
8.      return m
9.  def BSCNN(m, n):
10.     return m*n//USCLN(m,n)
11. def BSC3so(m,n,p):
12.     return BSCNN(BSCNN(m,n),p)
13. m,n,p = eval(input("Nhập 3 số tự nhiên cách nhau bởi dấu phảy
    m, n, p: "))
14. print("BSCNN là:",BSC3so(m,n,p))
```

11.21. Nhập số n và dãy số a_1, a_2, ..., a_n từ bàn phím. Tính:

a) Tổng các số của dãy trên.

b) Tìm phần tử (chỉ số) tương ứng với số lớn nhất và nhỏ nhất của dãy trên.

```
# 11.21
A = []
n = int(input("Nhập số tự nhiên n: "))
for k in range(1,n+1):
    a = float(input("Nhập số thứ "+str(k)+": "))
    A.append(a)
s = 0
for a in A:
    s = s + a
jmin = 0
jmax = 0
Min = A[jmin]
Max = A[jmax]
for k in range(n):
    if A[k] < Min:
        jmin = k
        Min = A[k]
    if A[k] > Max:
        jmax = k
        Max = A[k]
print(A)
print("Tổng các số đã nhập là:",s)
print("Chỉ số và số lớn nhất:", jmax, Max)
print("Chỉ số và số nhỏ nhất:", jmin, Min)
```

11.22. Nhập số n và dãy số a_1, a_2, ..., a_n từ bàn phím. Tìm một dãy con liên tục cực đại đơn điệu tăng của dãy số trên.

```
# 11.22
A = []
n = int(input("Nhập số tự nhiên n: "))
for k in range(1,n+1):
    a = int(input("Nhập số thứ "+str(k)+": "))
    A.append(a)
imax = 0
lenmax = 1
icurr = 0
lencurr = 1
for k in range(n-1):
    if A[k] < A[k+1]:
        lencurr = lencurr + 1
    else:
        if lenmax < lencurr:
            lenmax = lencurr
            imax = icurr
        icurr = k + 1
        lencurr = 1
print(A)
print("Dãy con đơn điệu tăng cực đại là:")
```

```
22. print(A[imax:imax + lenmax])
```

11.23. Tương tự bài 11.22. Tìm một dãy con liên tục cực đại các số > 0 của dãy trên.

```
1.  # 11.23
2.  A = []
3.  n = int(input("Nhập số tự nhiên n: "))
4.  for k in range(1,n+1):
5.      a = int(input("Nhập số thứ "+str(k)+": "))
6.      A.append(a)
7.  imax = 0
8.  lenmax = 1
9.  icurr = 0
10. lencurr = 1
11. for k in range(n-1):
12.     if A[k] > 0 and A[k+1] > 0:
13.         lencurr = lencurr + 1
14.     elif A[k] < 0 and A[k+1] >0:
15.         icurr = k+1
16.         lencurr = 1
17.     elif A[k] > 0 and A[k+1] < 0:
18.         if lenmax < lencurr:
19.             lenmax = lencurr
20.             imax = icurr
21. if lenmax < lencurr:
22.     lenmax = lencurr
23.     imax = icurr
24. print(A)
25. print("Dãy con đơn điệu tăng cực đại gồm các số > 0 là:")
26. print(A[imax:imax + lenmax])
```

11.24. Nhập số n từ bàn phím, tìm và in ra tất cả các ước số nguyên tố của số n.

Bài tập yêu cầu in ra các ước nguyên tố của số n, nhưng mỗi ước đó chỉ in ra đúng 1 lần.

```
1.  # 11.24
2.  n = int(input("Nhập số tự nhiên n: "))
3.  k = 2
4.  while n > 1:
5.      while n%k != 0:
6.          k = k + 1
7.      if n%k == 0:
8.          print(k, end = " ")
9.          while n%k == 0:
10.             n = n//k
```

11.25. Nhập 3 số m, n, p từ bàn phím. tìm và in ra tất cả các ước số chung của 3 số trên.

```
1.  # 11.25
2.  m,n,p = eval(input("Nhập 3 số tự nhiên cách nhau bởi dấu phảy
    m, n, p: "))
3.  count = 0
4.  for k in range(1,min(m,n,p)+1):
5.      if m%k == 0 and n%k == 0 and p%k == 0:
6.          count = count + 1
7.  print("Số lượng các ước chung của",m,n,p,"là:",count)
```

11.26. Số hoàn hảo là số = tổng các ước số thực sự của số đó. Tìm tất cả các số hoàn hảo nhỏ hơn 10000.

```
# 11.26 Tìm các số hoàn hảo
def perfect(n):
    s = 0
    for k in range(1,n//2+1):
        if n%k == 0:
            s = s + k
    if n == s:
        return True
    else:
        return False
n = int(input("Nhập số tự nhiên: "))
print("Các số hoàn hảo:",end = " ")
for k in range(1,n):
    if perfect(k):
        print(k,end = " ")
```

11.27. Nhập số n từ bàn phím, tìm:

a) Số nguyên tố nhỏ nhất > n.

b) Số nguyên tố lớn nhất < n.

c) Số nguyên tố nhỏ nhất > n/2.

d) Số nguyên tố lớn nhất < n/2.

```
# 11.27
def nguyento(n):
    k = 2
    while n%k != 0 and k < n:
        k = k + 1
    if k == n:
        return True
    else:
        return False
def NextPrime(n):
    n = n+1
    while not nguyento(n):
        n = n + 1
    return n
def PrevPrime(n):
    n = n - 1
    while not nguyento(n):
        n = n - 1
    return n

n = int(input("Nhập số tự nhiên n: "))
print("Số nguyên tố nhỏ nhất > n:",NextPrime(n))
print("Số nguyên tố lớn nhất < n:",PrevPrime(n))
print("Số nguyên tố nhỏ nhất > n/2:",NextPrime(n//2))
print("Số nguyên tố lớn nhất < n/2:",PrevPrime(n//2))
```

11.28. Nhập từ bàn phím số n, hãy viết n trong hệ nhị phân.

```
# 11.28
```

```
# Chuyển đổi số tự nhiên --> số nhị phân
binary = ["0","1"]
n = int(input("Nhập số tự nhiên: "))
def int_to_bin(n):
    bstr = ""
    while n > 0:
        n, r = divmod(n, 2)
        bstr = binary[r] + bstr
    return bstr
print("Biểu diễn nhị phân của",n,"là:",int_to_bin(n))
```

11.29. Nhập từ bàn phím số n, hãy viết n trong hệ hex (hệ đếm 16).

```
# 11.29. Chuyển đổi số tự nhiên --> số hex
hexa =
["0","1","2","3","4","5","6","7","8","9","A","B","C","D","E",
"F"]
n = int(input("Nhập số tự niên: "))
def int_to_hex(n):
    shex = ""
    while n > 0:
        n, r = divmod(n, 16)
        shex = hexa[r] + shex
    return shex
print("Biểu diễn hex của",n,"là:",int_to_hex(n))
```

11.30. Nhập số n và dãy số a_1, a_2, ..., a_n từ bàn phím. Tìm ra các phần tử cực trị địa phương của dãy này (phần tử cực trị địa phương nếu nó cùng lớn hơn hoặc nhỏ hơn 2 số bên cạnh).

```
# 11.30
A = [] # dãy gốc
B = [] # các phần tử cực trị địa phương
I = [] # chỉ số tương ứng
n = int(input("Nhập số tự nhiên n: "))
for k in range(1,n+1):
    a = int(input("Nhập số thứ "+str(k)+": "))
    A.append(a)
for k in range(2,len(A)-1):
    if A[k-1] < A[k] > A[k+1] or A[k-1] > A[k] < A[k+1]:
        B.append(A[k])
        I.append(k)
print(A)
print("Các phần tử cực trị địa phương là:")
print(B)
```

11.31. Viết hàm số **PrevPrime(n)** có ý nghĩa như sau:

- Tham số là số tự nhiên n.

- Hàm trả lại số nguyên tố lớn nhất nhỏ hơn hoặc bằng n. Nếu n = 1 thì trả lại False.

Bài tập này gần tương tự bài tập 11.7.

```
# 11.31
def nguyento(n):
    k = 2
    while n%k != 0 and k < n:
        k = k + 1
```

```
    if k == n:
        return True
    else:
        return False
def PrevPrime(n):
    k = n
    while not nguyento(k):
        k = k - 1
    return k
```

11.32. Viết chương trình thực hiện công việc sau:

- Yêu cầu người dùng liên tục nhập các số nguyên từ bàn phím. Quá trình nhập chỉ kết thúc khi người dùng nhấn phím ENTER khi chưa gõ bất cứ phím nào.

- Các số đã nhập sẽ được lưu vào một dãy, sau đó thể hiện trên màn hình.

Chương trình có thể đơn giản chỉ 1 lệnh như sau:

```
while input("Nhập một giá trị bất kỳ: "):
    pass
```

11.33. Viết chương trình thực hiện công việc sau:

- Yêu cầu người dùng liên tục nhập các số nguyên khác nhau đôi một từ bàn phím. Quá trình nhập chỉ kết thúc khi người dùng nhấn phím ENTER khi chưa gõ bất cứ phím nào.

- Nếu người dùng nhập một số đã được nhập từ trước thì thông báo ngay "Số này bạn đã nhập rồi" và yêu cầu nhập tiếp.

- Khi kết thúc chương trình in các số đã nhập (khác nhau) ra màn hình.

```
# 11.33
A = []
No_error = 1
while Noerror:
    No_error = input("Nhập số nguyên: ")
    if No_error:
        num = int(No_error)
        if num in A:
            print("Số này bạn đã nhập rồi, hãy nhập số khác.")
        else:
            A.append(num)
print("Các số đã nhập là:")
print(*A)
```

11.34. Khi viết lệnh này kết quả sẽ là gì?

```
x = 2 < 1 or print("hi")
```

Kết quả sẽ in ra "hi" và trả lại x giá trị **None**.

11.35. Khi viết lệnh này kết quả sẽ là gì?

```
x = print() or print("hi")
```

Kết quả sẽ xuống dòng, in ra chữ "hi" và trả lại x giá trị **None**.

11.36. Khi viết lệnh này kết quả sẽ là gì?

```
x = print("ha") or print("hi")
```

Kết quả sẽ in ra chữ "ha"", "hi" và trả lại x giá trị **None**.

11.37. Khẳng định sau đúng hay sai? Nêu lý do và giải thích.

Một mệnh đề logic (mệnh đề có các toán từ **and**, **or**, **not**) có thể không xác định được ngay giá trị vì còn phụ thuộc vào tương tác với người sử dụng.

Mệnh đề trên đúng.

Ví dụ xét lệnh sau:

```
b = 2 > 1 and input("Nhập 1 số: ")
```

Giá trị biến nhớ b sẽ phụ thuộc vào việc người dùng nhập 1 số từ bàn phím.

Chương 12: Đệ quy

12.1. Hàm f(n) được định nghĩa như sau:

```
def f(n):
    if n == 0:
        return 1
    else:
        return n + f(n-1)
```

Tính f(10).

a) 48

b) 50

c) 56

d) 94

Đáp án đúng: c)

12.2. Viết hàm đếm thời gian từ 1 trở đi, mỗi lần đếm là 1 giây. Đồng hồ thời gian sẽ dừng lại sau đúng 2 phút (tức sau 120 giây).

Hàm có thể được định nghĩa như sau:

```
def time(n = 1):
    if n <= 120:
        print(n)
        time(n+1)
```

12.3. Hàm số sau thực hiện công việc gì?

```
1.  def insertNum(n, A, k):
2.      newlist = []
3.      if k == 0:
4.          newlist = [n] + A
5.      elif k == len(A) + 1:
6.          newlist = A + [n]
7.      else:
8.          newlist = A[ : k] + [n] + A[k: ]
9.      return newlist
```

Trả lời: Hàm insertNum(n, A, k) sẽ chèn số n vào dãy A tại vị trí k.

Thực chất hàm này chính là hàm A.insert(k, n).

12.4. Hàm số sau thực hiện công việc gì:

```
1.  def insertALL(n, A):
2.      newlist = []
3.      for k in range(len(A) + 1):
4.          newlist.append(insertNum(n,A,k))
5.      return newlist
```

Chú ý: hàm này sử dụng hàm insertNum() trong bài tập 12.3.

Trả lời: Hàm này sẽ trả lại một dãy (list) mới bao gồm len(A) + 1 phần tử, mỗi phần tử đều là dãy thu được từ A bằng cách chèn số n vào lần lượt các vị trí 0, 1, 2, ..., len(A).

Ví dụ nếu A = [2, 5], n = 1 thì hàm insert(n, A) sẽ trả lại dãy:

```
[[1, 2, 5], [2, 1, 5], [2, 5, 1]]
```

12.5. Thuật toán sinh hoán vị ngẫu nhiên **Minimal Change**.

Yêu cầu nhập vào từ bàn phím số tự nhiên n, thể hiện trên màn hình tất cả các hoán vị của dãy [1, 2, ..., n]. Chương trình sau là phần chương trình lõi chính của thuật toán này.

Viết lại chương trình sau và trả lời các câu hỏi bên dưới.

```
def insertNum(n, A, k):
    newlist = []
    if k == 0:
       newlist = [n] + A
    elif k == len(A) + 1:
        newlist = A + [n]
    else:
        newlist = A[ : k] + [n] + A[k: ]
    return newlist
def insertALL(n, A):
    newlist = []
    for k in range(len(A) + 1):
        newlist.append(insertNum(n,A,k))
    return newlist
def permutation(n):
    if n == 1:
        return [1]
    elif n == 2:
        return [[1,2], [2,1]]
    else:
        new = []
        A = permutation(n-1)
        for pe in A:
            new = new + insertALL(n,pe)
        return new
A = permutation(4)
print(A)
```

a) Thực hiện hàm số với n = 1, 2, 3 để xem kết quả và dự đoán chức năng chính của hàm **permutation()**.

b) Hàm permutation(2) trả lại list [[1,2], [2,1]], em có nhận xét gì về kết quả này.

c) Chứng minh rằng hàm **permutation(n)** là hàm sinh các hoán vị của dãy 1, 2, 3, ..., n.

d) Kết quả hàm **permutation(n)** trả lại dữ liệu gì?

e) Dựa vào đoạn chương trình trên viết lại chương trình mô tả yêu cầu của bài toán này.

Trả lời câu a).

Kết quả chạy hàm permutation(n) với n = 1, 2, 3 thu được các kết quả sau:

```
[1]
[[1, 2],[2, 1]]
[[3, 1, 2],[1, 3, 2],[1, 2, 3],[3, 2, 1],[2, 3, 1],[2, 1, 3]]
```

Dự đoán: Hàm permutation sẽ sinh tất cả các hoán vị của dãy 1, 2, ..., n.

Câu b).

Nhận xét: Dãy [[1,2], [2,1]] thu được từ dãy [1] bằng cách chèn số 2 vào các vị trí trước và sau số *1 của dãy [1].*

Câu c).

Từ cách thiết lập hàm đệ quy permutation(n), chúng ta có nhận xét:

- Với n = 1, hàm trả lại dãy [1].

- Với n = 2, hàm trả lại [[1,2],[2,1]]

- Với n tổng quát, hàm permutation(n) sẽ dựa trên kết quả của permutation(n-1) và đưa vào mỗi dãy của permutation(n-1) đúng n lần bằng cách chèn số n lần lượt vào các vị trí của mỗi dãy của permutation(n-1).

Từ các nhận xét trên dễ dàng dùng quy nạp để chứng minh hàm permutation(n) sẽ trả lại tất cả các hoán vị của dãy 1, 2, ..., n.

Câu d).

Hàm permutation(n) trả lại một dãy (list) mà mỗi phần tử của nó là một list - dãy hoán vị của 1, 2, ..., n.

Câu e).

Chương trình như sau:

```
1.  # 12.5e
2.  n = int(input("Nhập số tự nhiên n: "))
3.  def insertNum(n, A, k):
4.      newlist = []
5.      if k == 0:
6.          newlist = [n] + A
7.      elif k == len(A) + 1:
8.          newlist = A + [n]
9.      else:
10.         newlist = A[ : k] + [n] + A[k: ]
11.     return newlist
12. def insertALL(n, A):
13.     newlist = []
14.     for k in range(len(A) + 1):
15.         newlist.append(insertNum(n,A,k))
16.     return newlist
17. def permutation(n):
18.     if n == 1:
19.         return [1]
20.     elif n == 2:
21.         return [[1,2], [2,1]]
22.     else:
23.         new = []
24.         A = permutation(n-1)
25.         for pe in A:
26.             new = new + insertALL(n,pe)
27.         return new
28. A = permutation(n)
29. for h in A:
30.     print(*h)
```

Chú ý: Hàm insertNum() trong chương trình trên có thể viết lại đơn giản hơn như sau:

```
def insertNum(n, A, k):
    newlist = A.copy()
    neslist.insert(n,k)
    return newlist
```

12.6. Giải thích chương trình sau.

```
1.  from random import *
2.  n = int(input("Nhập số tự nhiên n: "))
3.  A = []
4.  for i in range(1, n+1):
5.      A.append(i)
6.  C = []
7.  while A != []:
8.      k = choice(range(len(A)))
9.      a = A[k]
10.     C.append(a)
11.     A.remove(a)
12. print(C)
```

Trả lời.

Chương trình yêu cầu nhập từ bàn phím số tự nhiên n và thiết lập một hoán vị ngẫu nhiên của các số 1, 2, ..., n, kết quả đưa vào dãy C. Cách thiết lập C như sau:

- Thiết lập dãy A = [1, 2, ..., n].

- Lần lượt lấy 1 phần tử ngẫu nhiên từ A, đưa vào C, sau đó xóa phần tử này khỏi dãy A.

12.7. Cho trước số tự nhiên n nhập từ bàn phím.

Viết chương trình tìm tất cả các khai triển:

$N = x_1 + x_2 + ... + x_k$, với x_i là số tự nhiên.

Chương trình cần liệt kê tất cả các bộ nghiệm:

$(x_1, x_2, ..., x_k)$

Ví dụ:

N = 1	1 = 1
N = 2	2 = 1 + 1 2 = 2
N = 3	3 = 1 + 1 + 1 3 = 1 + 2 3 = 2 + 1 3 = 3

Gợi ý:

Bộ dữ liệu được thiết lập là dãy (list) A có dạng, chú ý rằng chỉ số sẽ đánh số từ 0.

$A = [x_0, x_1, ..., x_k]$

Lời giải được thực hiện đệ quy bởi hàm **gpt(A,n,p)** như sau:

Lời gọi ban đầu là gpt(A,N, 0).

$N = x_1 + x_2 + ... + x_p + x_{p+1} + ... + x_k$

Lời gọi hàm **gpt(A,n,p)** bắt đầu từ vị trí này. Hàm sẽ được thực hiện đệ quy như sau:

```
for k in range(1, n+1):
    if len(A) > p:
       A = Copy(A,p) # xóa các phần tử sau p
    A.append(k)
    gpt(A,n-k, p+1)
```

Chương trình chính sẽ có dạng đệ quy như sau:

```
1.  def gpt(A,n,p):
2.      if n == 0:
3.          print(A)
4.          return
5.      else:
6.          for k in range(1, n+1):
7.            if len(A) > p:
8.               A = Copy(A,p)
9.            A.append(k)
10.           gpt(A,n-k, p+1)
11.     return
12. # main chương trình chính
13. n = int(input("Nhập số n: "))
14. A = []
15. gpt(A,n,0)
```

Hàm này có chức năng xóa các phần tử của dãy A tính từ vị trí p.

Lời giải đầy đủ.

```
1.  # 12.7
2.  # Giải phương trình n = x0+x1+x2+...+xk với xi>0.
3.  def Copy(A,k):
4.      Temp = []
5.      for i in range(k):
6.          Temp.append(A[i])
7.      return Temp
8.  def ShowSolution(A):
9.      print(n,"=",end = " ")
10.     for k in range(len(A)):
11.         if k < len(A)-1:
12.             print(A[k],"+",end = " ")
13.         else:
14.             print(A[k])
15. def gpt(A,n,p):
16.     if n == 0:
17.         ShowSolution(A)
18.         return
19.     else:
20.         for k in range(1, n+1):
21.             if len(A) > p:
22.                 A = Copy(A,p)
23.             A.append(k)
24.             gpt(A,n-k, p+1)
25.     return
```

```
# Chương trình chính
n = int(input("Nhập số n: "))
A = []
gpt(A,n,0)
```

Kết quả chạy chương trình, ví dụ với n = 3 sẽ như sau:

```
Nhập số n: 3
3 = 1 + 1 + 1
3 = 1 + 2
3 = 2 + 1
3 = 3
```

12.8. Giải quyết bài toán trong bài tập 12.7 theo một gợi ý khác.

List A được thiết lập một lần từ đầu với độ dài cố định (khá lớn). Ví dụ A được thiết lập ban đầu bao gồm toàn các phần tử 0 như sau:

```
MAX = 10000
A = []
for i in range(MAX):
    A.append(0)
```

Thủ tục đệ quy chính có dạng tương tự như sau:

```
def gpt(A,n,p):
    if n == 0:
        Show(A)
        return
    else:
        for k in range(1, n+1):
          if len(A) > p:
              ..... # xóa các phần tử sau chỉ số p
          A[p] = k
          gpt(A,n-k, p+1)
    return
```

Em hãy hoàn thiện chương trình trên và hoàn thiện yêu cầu của bài tập này.

Lời giải.

```
# 12.8
# Giải phương trình: x1 + x2 + ... + xk = n, với xj > 0
def Copy(A,p):
    if len(A) <= p:
        return A
    else:
        return A[ : p]
def gpt(A,n,p):
    if n == 0:
        print(A)
        return
    else:
        for k in range(1, n+1):
            if len(A) > p:
                A = Copy(A,p)
            A.append(k)
            gpt(A,n-k, p+1)
    return
# main chương trình chính
```

```
20. n = int(input("Nhập số n: "))
21. A = []
22. gpt(A,n,0)
```

Ví dụ kết quả chương trình trên với n = 4 là:

```
Nhập số n: 4
[1, 1, 1, 1]
[1, 1, 2]
[1, 2, 1]
[1, 3]
[2, 1, 1]
[2, 2]
[3, 1]
[4]
```

12.9. Xét bài toán sau tổng quát hơn bài toán trong các bài tập trên:

Cho trước số N và k số tự nhiên cho trước m_1, m_2, ..., m_k.

Cần tìm tất cả các nghiệm nguyên (lớn hơn hoặc bằng 0) của phương trình:

$N = x_1m_1 + x_2m_2 + ... + x_km_k$ với $x_i \geq 0$

Chương trình cần liệt kê trên màn hình tất cả các nghiệm của phương trình trên. Nếu không có nghiệm, chương trình sẽ thông báo vô nghiệm.

Ví dụ với k = 3.

$m_1 = 1$, $m_2 = 2$, $m_3 = 3$, N = 5, chương trình cần liệt kê các nghiệm như sau:

```
0 1 1
1 2 0
2 0 1
3 1 0
5 0 0
```

Giải bài toán trên khi bộ dữ liệu nghiệm được lưu trong các dãy (list) có đúng k phần tử dạng:

$A = [a_1, a_2, ..., a_k]$.

```
1.  # 12.9
2.  # Giải phương trình: x1m1 + x2m2 + ... + xkmk = n
3.  K = 3
4.  M = [1, 2, 3]
5.  A = []
6.  def init(k):
7.      return [0]*k
8.  def Clear(A,p):
9.      for i in range(p,K):
10.         A[i] = 0
11. def Show(A):
12.     for i in range(K):
13.         print(A[i], end = " ")
14.     print()
15. def gpt(A,n,p):
16.     if n == 0:
17.         Show(A)
18.         return
```

```
    elif p == K-1:
        if n%M[p] == 0:
            A[p] = n // M[p]
            Show(A)
            return
        else:
            return
    else:
        a, b = divmod(n, M[p])
        for k in range(a+1):
            Clear(A,p)
            A[p] = k
            gpt(A,n-k*M[p], p+1)
    return
# Chương trình chính
n = int(input("Nhập số n: "))
A = init(K)
gpt(A,n,0)
```

Ví dụ với n = 5, chúng ta có kết quả sau:

```
Nhập số n: 5
0 1 1
1 2 0
2 0 1
3 1 0
5 0 0
```

12.10. Mở rộng yêu cầu của bài tập 9. Yêu cầu chương trình thông báo tổng số nghiệm tìm được trước khi thể hiện tất cả các nghiệm trên màn hình.

Ví dụ đầu ra của chương trình như sau.

```
Tổng số nghiệm: 5
0 1 1
1 2 0
2 0 1
3 1 0
5 0 0
```

Lời giải.

```
# 12.10
# Giải phương trình: x1m1 + x2m2 + ... + xkmk = n
# Tính số nghiệm và sau đó in ra các nghiệm.
K = 3
M = [1, 2, 3]
Solution = []
A = []
def init(k):
    return [0]*k
def Clear(A,p):
    for i in range(p,K):
        A[i] = 0
def gpt(A,n,p):
    if n == 0:
        temp = A.copy()
        Solution.append(temp)
```

```
            return
        elif p == K-1:
            if n%M[p] == 0:
                A[p] = n // M[p]
                temp = A.copy()
                Solution.append(temp)
                return
            else:
                return
        else:
            a, b = divmod(n, M[p])
            for k in range(a+1):
                Clear(A,p)
                A[p] = k
                gpt(A,n-k*M[p], p+1)
    return
# Chương trình chính
n = int(input("Nhập số n: "))
A = init(K)
gpt(A,n,0)
print("Phương trình có",len(Solution),"nghiệm")
for s in Solution:
    print(*s)
```

12.11. Viết các hàm số sau sử dụng lập trình đệ quy.

a) Hàm **S(n)** là tổng các số tự nhiên ≤ n.

S(n) = 1 + 2 + ... + n.

b) Hàm **SEven(n)** là tổng các số tự nhiên chẵn ≤ n.

c) Hàm **SOdd(n)** là tổng các số tự nhiên lẻ ≤ n.

Câu a).

```
def S(n):
    if n == 0:
        return 0
    else:
        return n + S(n-1)
```

Câu b).

```
def Seven(n):
    if n == 0:
        return 0
    else:
        if n%2 == 0:
            return n + Seven(n-1)
        else:
            return Seven(n-1)
```

Câu c).

```
def Sodd(n):
    if n == 0:
        return 0
    else:
        if n%2 == 1:
```

```
            return n + Sodd(n-1)
        else:
            return Sodd(n-1)
```

12.12. Hàm sau đây thực hiện công việc gì?

```
def Rev(A):
    if A == []:
        return []
    kqLast = Rev(A[1 : ])
    item = A[0 : 1]
    return kqLast + item
```

Hàm trả về dãy ngược lại của dãy A.

12.13. Dựa trên bài tập 12, viết hàm tương tự nhưng áp dụng cho xâu ký tự.

Trả lời.

```
def Rev(A):
    if A == "":
        return ""
    kqLast = Rev(A[1 : ])
    item = A[0 : 1]
    return kqLast + item
```

12.14. Viết các hàm số sau sử dụng lập trình đệ quy.

a) Hàm Exp(x, n) tính lũy thừa x^n với x là số thực bất kỳ, n là số tự nhiên.

b) Hàm FacEven(n) tính tích các số tự nhiên chẵn ≤ n.

c) Hàm FacOdd(n) tính tích các số tự nhiên lẻ ≤ n.

Câu a).

```
def Exp(x,n):
    if n == 0:
        return 1
    else:
        return x*Exp(x,n-1)
```

Câu b).

```
def FacEven(n):
    if n <= 1:
        return 1
    else:
        if n%2 == 0:
            return n*FacEven(n-1)
        else:
            return FacEven(n-1)
```

Câu c).

```
def FacOdd(n):
    if n <= 1:
        return 1
    else:
        if n%2 == 1:
            return n*FacOdd(n-1)
        else:
            return FacOdd(n-1)
```

12.15. Viết chương trình tính các tổng sau bằng phương pháp đệ quy.

a) $S = 1 + 2 + ... + n$

b) $S = 1^3 + 2^3 + ... + n^3$

c) $S = \frac{1}{1.2} + \frac{1}{2.3} + \cdots . + \frac{1}{n(n+1)}$

d) $S = \frac{1}{1} + \frac{1}{1.2} + \frac{1}{1.2.3} + \cdots . + \frac{1}{n!}$

Câu a)

```
def S(n):
    if n == 1:
        return 1
    else:
        return n + S(n-1)
```

Câu b).

```
def S(n):
    if n == 1:
        return 1
    else:
        return n*n*n + S(n-1)
```

Câu c)

```
def S(n):
    if n == 1:
        return 0.5
    else:
        return 1/(n*(n+1)) + S(n-1)
```

Câu d).

```
def S(n):
    if n == 1:
        return 1
    else:
        F = 1
        for k in range(1,n+1):
            F = F*k
        return 1/F + S(n-1)
```

12.16. Cho đoạn chương trình sau:

```
1.  def LenNum(str):
2.      if str == "":
3.          return 0
4.      elif len(str) == 1:
5.          if ord(str[0]) >= 48 and ord(str[0]) <= 57:
6.              return 1
7.          else:
8.              return 0
9.      else:
10.         if ord(str[0]) >= 48 and ord(str[0]) <= 57:
11.             return 1 + LenNum(str[1 : ])
```

```
12.         else:
13.             return LenNum(str[1 : ])
```

a) Chương trình trên có sử dụng đệ quy hay không?

b) Nêu ý nghĩa của hàm số LenNum(str) trong chương trình trên.

c) Chương trình trên có một vài đoạn code chưa tối ưu, bạn có thể viết rút gọn lại được không?

Trả lời:

a) Có.

b) Hàm LenNum(str) đếm số các ký tự là số trong xâu str.

c) Có thể viết lại hàm này đơn giản hơn như sau:

```
def LenNum(Str):
    if Str == "":
        return 0
    else:
        if "0" <= Str[0] <= "9":
            return 1 + LenNum(Str[1 : ])
        else:
            return LenNum(Str[1 : ])
```

12.17. Viết các chương trình sau đều yêu cầu nhập từ bàn phím một xâu ký tự và tính:

a) Tính số các ký tự là số trong xâu (các ký tự 0, 1, 2, ..., 9).

b) Tính số các ký tự là chữ cái (chữ hoa hoặc thường) trong xâu.

c) Tính số các ký tự không là số và chữ trong xâu.

Yêu cầu viết các chương trình trên bằng kỹ thuật đệ quy.

Chúng ta sẽ viết lời giải chỉ là các hàm đệ quy để tính theo yêu cầu từng câu.

Câu a).

```
def LenNum(Str):
    if Str == "":
        return 0
    else:
        if "0" <= Str[0] <= "9":
            return 1 + LenNum(Str[1 : ])
        else:
            return LenNum(Str[1 : ])
```

Câu b).

```
def LenAlphabet(Str):
    if Str == "":
        return 0
    else:
        if "a" <= Str[0] <= "z" or "A" <= Str[0] <= "Z":
            return 1 + LenAlphabet(Str[1 : ])
        else:
            return LenAlphabet(Str[1 : ])
```

Câu c).

```
def LenSpecial(Str):
    if Str == "":
        return 0
    else:
        if "a" <= Str[0] <= "z" or "A" <= Str[0] <= "Z" or "0" <=
Str[0] <= "9":
            return LenSpecial(Str[1 : ])
        else:
            return 1 + LenSpecial(Str[1 : ])
```

12.18. Viết phương trình tìm tất cả các nghiệm của phương trình:

$N = x_1 + x_2 + x_3$

với điều kiện $0 \le N \le 30$, $0 \le x_1, x_2, x_3 \le 9$.

```
# 12.18
# Giải phương trình x1 + x2 + x3 = N, 0<=xi<=9
p = [0,0,0]
M = 3
N = 10
def gpt(N,m):
    if m == 0:
        if N <= 9:
            p[m] = N
            print(p)
            return
        else:
            return
    else:
        if N > 0:
            for k in range(min(N,9)+1):
                p[m] = k
                gpt(N-k,m-1)
        else:
            print(p)
            return
gpt(N,M-1)
```

12.19. Tổng quát bài tập 18, viết chương trình tìm tất cả các nghiệm của phương trình:

$N = x_1 + x_2 + ... + x_m$

với điều kiện:

$N \ge 0$, $0 \le x_1, x_2, x_3, ..., x_m \le 9$.

```
# 12.19
# Giải phương trình x1 + x2 + ... + xm = N, 0<=xi<=9
p = []
M = 5
N = 10
def init(M):
    return [0]*M
def gpt(N,m):
    if m == 0:
        if N <= 9:
```

```
                p[m] = N
                print(p)
                return
            else:
                return
        else:
            if N > 0:
                for k in range(min(N,9)+1):
                    p[m] = k
                    gpt(N-k,m-1)
            else:
                print(p)
                return
p = init(M)
gpt(N,M-1)
```

12.20. Viết chương trình giải phương trình tổng quát của bài tập 19, hạn chế các nghiệm x_i theo giới hạn cho trước:

$0 \le x_i \le C_i$, với $C_1, C_2, ..., C_m$ là các hằng số cho trước.

```
# 12.20
# Giải phương trình x1 + x2 + ... + xm = N, 0<=xi<=Ci
C = [3,1,6,10]
M = 4
N = 15
def init(M):
    return [0]*M
def gpt(N,m):
    if m == 0:
        if N <= C[m]:
            p[m] = N
            print(p)
            return
        else:
            return
    else:
        if N > 0:
            for k in range(min(N,C[m])+1):
                p[m] = k
                gpt(N-k,m-1)
        else:
            print(p)
            return
p = init(M)
gpt(N,M-1)
```

Chương 13: Kiểu dữ liệu từ điển

13.1. Giả sử d là một biến dữ liệu kiểu từ điển cho trước. Khi đó lệnh:

```
d[1] = "Hà"
```

sẽ thực hiện công việc gì?

a) Tạo thêm một bản ghi (cặp key - value) mới <1 : "Hà"> và đưa vào phần tử đầu tiên của từ điển.

b) Tạo thêm một bản ghi (cặp key - value) mới <1 : "Hà"> và đưa vào phần tử cuối cùng của từ điển.

c) Nếu trong từ điển đã có key (khóa) = 1 rồi thì cập nhật giá trị mới "Hà" vào khóa này, ngược lại thì tạo thêm một bản ghi (cặp key - value) mới <1 : "Hà"> và đưa vào phần tử cuối cùng của từ điển.

d) Lệnh báo lỗi.

Đáp án đúng: c)

13.2. Giả sử từ điển d được định nghĩa như sau:

d = {1: 'Bình', 3: 'Thuận', 2: 'Quang'}

Khi đó lệnh:

```
list(d)
```

trả lại kết quả gì?

a) Dãy (list) các cặp key-value của từ điển.

b) Dãy (list) các key của từ điển.

c) Dãy (list) các value của từ điển.

Đáp án đúng: b) Dãy (list) các key của từ điển.

13.3. Giả sử từ điển d được định nghĩa như sau:

d = {1: 'Bình', 3: 'Thuận', 2: 'Quang'}

Khi đó lệnh:

```
print(d["Quang"])
```

trả lại kết quả gì?

a) 2

b) "Quang"

c) [2 : "Quang"]

d) Chương trình báo lỗi.

Đáp án đúng: d)

13.4. Giả sử từ điển d được định nghĩa như sau:

d = {1: 'Bình', 3: 'Thuận', 2: 'Quang'}

Khi đó lệnh:

```
d.get(0)
```

trả lại kết quả gì?

a) Chương trình báo lỗi.

b) None

c) Trả lại một kết quả khác với 2 kết quả trên.

Đáp án đúng: b)

13.5. Giả sử từ điển d được định nghĩa như sau:

d = {1: 'Bình', 3: 'Thuận', 2: 'Quang'}

Khi đó lệnh:

```
d.get(0, "Hà")
```

trả lại kết quả gì?

a) Chương trình báo lỗi.

b) None

c) "Hà"

d) Một kết quả khác.

Đáp án đúng: c)

13.6. Viết chương trình tạo và nhập dữ liệu từ điển từ một text file như sau:

a) Bộ dữ liệu từ điển bao gồm Key là họ tên học sinh, Value là điểm số.

b) Tệp dữ liệu có dạng sau:

- Dòng đầu tiên ghi số N là số học sinh trong lớp.

- N dòng tiếp theo là dữ liệu điểm học sinh, mỗi dòng bao gồm Họ tên học sinh và giá trị điểm số, cách nhau bởi dấu phẩy.

Ví dụ khuôn dạng một tệp như vậy:

```
5
Nguyen Hoa Binh, 9.5
Bui Viet Ha, 8
Pham Quang Vinh, 7.5
Nguyen Thi Hoa, 10
Truong Van Quang, 8.5
```

Lời giải.

```
# 13.6
d = {}
fdata = "B13-6.dat"
f = open(fdata,"r")
n = int(f.readline())
for line in f:
    line = line.split(",")
    ten = line[0]
    diem= float(line[1])
    d[ten] = diem
f.close()
print(d)
```

Kết quả như sau:

```
{'Nguyen Hoa Binh': 9.5, 'Bui Viet Ha': 8.0, 'Pham Quang Vinh':
7.5, 'Nguyen Thi Hoa': 10.0, 'Truong Van Quang': 8.5}
```

13.7. Giả sử từ điển dic được định nghĩa như sau:

dic = {'b': 2, 'c' : 3, 'a' : 1}

Khi đó đoạn chương trình sau cho ta kết quả gì?

```
for x in sorted(dic):
    print(d[x], end = " ")
```

a) a b c

b) 0 1 2

c) Chương trình báo lỗi.

Đáp án đúng: a)

13.8. Giả sử từ điển dic được định nghĩa như sau:

dic = {'b': 2, 'c' :3, 'a' : 1}

Khi đó đoạn chương trình sau cho ta kết quả gì?

```
for x in sorted(dic):
    print(x, end = " ")
```

a) a b c

b) 0 1 2

c) Chương trình báo lỗi.

Đáp án đúng: a).

13.9. Cho đoạn chương trình sau:

```
dic = {1 : 'a', 2 : 'b', 4 : "d", 10 : "k"}
b = 0 in dic
```

b sẽ nhận giá trị nào?

a) True

b) False

c) Giá trị khác với True và False.

d) Chương trình báo lỗi.

Đáp án đúng: b).

13.10. Cho đoạn chương trình sau:

```
dic = {1 : 'a', 2 : 'b', 4 : "d", 10 : "k"}
b = 'a' in dic
```

b sẽ nhận giá trị nào?

a) True

b) False

c) Giá trị khác với True và False.

d) Chương trình báo lỗi.

Đáp án đúng: b).

13.11. Giả sử chúng ta đã có một dữ liệu từ điển thông tin điểm học sinh bao gồm cặp <key>-<value> là <Họ tên> - < Điểm số>. Ví dụ bộ dữ liệu gốc đã có:

dic = {"Bùi Việt Hà" : 8, "Nguyễn Thanh Bình" : 9.5}

Viết một chương trình có chức năng cập nhật điểm số học sinh vào bộ dữ liệu trên. Chương trình thực hiện như sau:

Chương trình yêu cầu nhập lần lượt 2 tham số:

1) Nhập họ tên học sinh.

2) Nhập điểm số.

Các thông tin trên sẽ được cập nhật vào bộ dữ liệu, hoặc tạo mới một bản ghi, hoặc cập nhật điểm của học sinh đã có trong danh sách. Chương trình sẽ đưa ra thông báo như sau:

Đã tạo mới một học sinh và nhập điểm.

hoặc

Đã cập nhật thông tin điểm của học sinh này.

```
1.  # 13.11
2.  dic = {"Bùi Việt Hà" : 8, "Nguyễn Thanh Bình" : 9.5}
3.  tenhs = input("Nhập họ tên học sinh: ")
4.  diem = float(input("Nhập điểm: "))
5.  if tenhs in dic:
6.      print("Đã cập nhật thông tin điểm của học sinh này.")
7.  else:
8.      print("Đã tạo mới một học sinh và nhập điểm.")
9.  dic[tenhs] = diem
10. print(dic)
```

13.12. Từ điển D được định nghĩa bao gồm Khóa là tên các bạn học sinh trong lớp, Value là cặp 2 giá trị số là điểm trung bình Văn và Toán. Ví dụ một bộ dữ liệu từ điển như vậy. Bộ dữ liệu điểm được ghi trong một list.

```
D = {"Hà" : [7, 9.5], "Bình" : [10, 8], "Hoa" : [5.5, 9]}
```

Viết chương trình nhập bộ dữ liệu từ điển trên như sau:

- Đầu tiên yêu cầu người dùng nhập tên học sinh.

- Tiếp theo yêu cầu nhập điểm Văn và Toán trên một dòng bởi 2 số, cách nhau bởi dấu phẩy.

- Tiếp theo thông báo đã cập nhật xong. Có 2 loại thông báo như sau:

- Thông báo loại 1: "Đã nhập điểm cho học sinh mới".

- Thông báo loại 2: "Đã cập nhật lại điểm cho học sinh trong danh sách".

```
1.  # 13.12
2.  D = {"Hà" : [7, 9.5], "Bình" : [10, 8], "Hoa" : [5.5, 9]}
3.  tenhs = input("Nhập họ tên học sinh: ")
4.  d1,d2 = eval(input("Nhập điểm Toán, Văn: "))
5.  if tenhs in D:
```

```
6.      print("Đã cập nhật thông tin điểm của học sinh này.")
7.  else:
8.      print("Đã tạo mới một học sinh và nhập điểm.")
9.  D[tenhs] = [d1,d2]
10. print(D)
```

13.13. Cho trước bộ dữ liệu điểm Văn, Toán của các bạn trong lớp em như trong bài tập 12, ví dụ một bộ dữ liệu như vậy:

```
D = {"Hà" : [7, 9.5], "Bình" : [10, 8], "Hoa" : [5.5, 9]}
```

Viết chương trình tính điểm trung bình của học sinh lớp em và thể hiện trên màn hình như sau:

```
Điểm trung bình:
Hà: 8.25
Bình: 9.0
Hoa: 7.25
```

Lời giải.

```
1.  # 13.13
2.  D = {"Hà" : [7, 9.5], "Bình" : [10, 8], "Hoa" : [5.5, 9]}
3.  def diemtb(diem):
4.      return(sum(diem)/len(diem))
5.  for ten,diem in D.items():
6.      print(ten,diemtb(diem))
```

13.14. Cho đoạn chương trình sau:

```
>>>D = {"Hà" : [7, 9.5], "Bình" : [10, 8], "An" : [6, 9]}
>>> Dic = sorted(D)
```

Khi đó Dic có kết quả gì?

a) Dic = {"Hà" : [7, 9.5], "Bình" : [10, 8], "An" : [6, 9]}

b) Dic = {"An" : [6, 9], "Hà" : [7, 9.5], "Bình" : [10, 8] }

c) Dic = {"An", "Hà", "Bình"}

d) Chương trình báo lỗi.

Đáp án đúng: d).

13.15. Cho trước 3 bộ dữ liệu từ điển các bạn học sinh lớp em. Cả 3 bộ dữ liệu này đều có khuôn dạng giống nhau với bộ <key - value> là <Tên học sinh - Điểm số>, nhưng đó là 3 bộ điểm số các môn Toán, Văn, Ngoại ngữ.

Viết chương trình cho trước 3 bộ từ điển trên là DToan, DVan, DNgoaingu. Cần tạo ra một bộ dữ liệu mới có tên DTB sẽ tính điểm trung bình của 3 môn trên.

Tuy nhiên do dữ liệu của từng môn có thể thiếu nên danh sách tên học sinh của các môn trên có thể thiếu, do vậy chương trình yêu cầu chỉ tính điểm trung bình cho các bạn học sinh có đủ điểm cả 3 môn. Chương trình sẽ bỏ qua không tính điểm trung bình của các bạn học sinh thiếu điểm của 1 trong 3 môn trên.

```
1.  # 13.15
2.  DToan = {"Hà": 7.0, "Bình": 8.5, "Sơn": 10,"Quang": 5.8,"Ngọc":
    5.0}
3.  DVan = {"Hà": 8.5, "Ngọc": 9.1,"Thuận": 9.0,"Sơn": 10,"Quang":
    9.8}
```

```
4. DNN = {"An": 10, "Vinh": 8.3, "Hà": 10, "Ngọc": 7.5,"Thuận":
   7.0,"Sơn":5.5,"Quang": 10}
5. for ten in DToan:
6.     if ten in DVan and ten in DNN:
7.         print(ten,end = " ")
8.         diem = (DToan[ten] + DVan[ten] + DNN[ten])/3
9.         print(round(diem,2))
```

13.16. Hoàn thiện chương trình ứng dụng đã có trong bài học như sau: Cho trước tệp văn bản Vanban.txt lưu trữ đoạn văn bản (có thể lớn bất kỳ). Chương trình sẽ đọc dữ liệu từ tệp văn bản này và in ra màn hình tần suất xuất hiện các từ đơn trong văn bản.

```
1. # 13.16
2. s = """This may seem extremely limiting, but a single bit is
3. enough to represent whether the answerto a question is yes or
4. no or a single bit can be used to tell us whether a logical
5. statement evaluates to either true or false"""
6. A = s.split()
7. dic = {}
8. for w in A:
9.   dic[w] = dic.get(w,0) + 1
10. for key in dic:
11.   print(key, dic[key])
```

13.17. Dữ liệu đầu vào như bài 16. Chương trình yêu cầu nhập từ bàn phím số tự nhiên n và in ra màn hình n từ có tần suất xuất hiện lớn nhất của văn bản. Hiển thị các từ và số lần xuất hiện của từ trong văn bản.

```
1. # 13.17
2. s = """This may seem extremely limiting, but a single bit is
3. enough to represent whether the answerto a question is yes or
4. no or a single bit can be used to tell us whether a logical
5. statement evaluates to either true or false"""
6. t = int(input("Nhập số t = tần suất mong đợi: "))
7. A = s.split()
8. dic = {}
9. for w in A:
10.     dic[w] = dic.get(w,0) + 1
11. for key in dic:
12.     if dic[key]>=t:
13.         print(key, dic[key])
```

13.18. Dữ liệu đầu vào như bài 16. Cho trước dãy (list) các từ khóa nào đó. Gọi dãy này là KeyWords. Viết chương trình thiết lập một từ điển dic với tập các key lấy từ KeyWords và giá trị là tần suất xuất hiện của các từ này trong văn bản đã cho.

```
1. # 13.18
2. s = """This may seem extremely limiting, but a single bit is
3. enough to represent whether the answerto a question is yes or
4. no or a single bit can be used to tell us whether a logical
5. statement evaluates to either true or false"""
6. K = ["bit","is","or","a","true","false"]
7. dic = {} # từ điển cần tìm
8. A = s.split()
9. d = {}
10. for w in A:
11.     d[w] = d.get(w,0) + 1
```

```
for w in K:
    dic[w] = d.get(w,0)
print(dic)
```

13.19. Cho trước từ điển D. Viết chương trình chuyển đổi dữ liệu của từ điển D sang dữ liệu dạng list với đầy đủ cả khóa và giá trị.

Ví dụ nếu D có dạng: {"Hà": 8, "Bình": 9.5, "Hoa": 10} thì list thu được sẽ phải là:

[["Hà", 8], ["Bình", 9.5], ["Hoa", 10]].

```
# 13.19
d = {"Hà": 8, "Bình": 9.5, "Hoa": 10}
A = []
for key, val in d.items():
    A.append([key,val])
```

13.20. Cho trước một số tệp văn bản. Các tệp này có thể được bổ sung vào chương trình bất cứ lúc nào. Viết chương trình dùng để khảo sát tần suất xuất hiện các từ khóa lặp nhiều nhất. Chương trình có các chức năng sau:

1) Nạp thêm các văn bản từ tệp vào bộ nhớ. Mỗi lần nạp văn bản, từ điển sinh bởi chương trình sẽ được cập nhật (xem chức năng 2).

2) Thiết lập một từ điển với 10 từ khóa xuất hiện nhiều nhất trong các văn bản đã khảo sát (đã nhập vào từ chức năng 1).

3) In danh sách 10 từ khóa hot nhất cùng với tần suất xuất hiện các từ này trong các văn bản đã khảo sát.

```
# 13.20
dic = {}
s = """sample text"""
A = s.split()
for w in A:
    dic[w] = dic.get(w,0) + 1
def Addtext():
        filename = input("Nhập tên text file: ")
        f = open(filename,"r")
        s = f.read()
        A = s.split()
        for w in A:
            dic[w] = dic.get(w,0) + 1
        f.close()
def Showhot():
        d = sorted(dic,key = dic.get,reverse = True)
        i = 1
        for key in d:
            print(i, key, dic.get(key))
            i = i + 1
            if i > 10:
                break
while True:
    print("""
    1. Bổ sung thêm text file
    2. In 10 từ khóa hot nhất
    3. Thoát.
    """)
```

```
    k = int(input("Nhập chức năng cần thực hiện (1,2,3): "))
    if k == 1:
        Addtext()
    if k == 2:
        Showhot()
    if k == 3:
        break
```

Chương 14: Kiểu dữ liệu tập hợp

14.1. Viết chương trình tạo tập hợp A bao gồm 5 phần tử ngẫu nhiên lấy từ dãy 0, 1, 2, 3, ..., 9.

```
1. # Bài 14.1
2. from random import choice
3. C = [0,1,2,3,4,5,6,7,8,9]
4. A = set()
5. for i in range(5):
6.     k = choice(C)
7.     A.add(k)
8.     C.remove(k)
9. print(A)
```

14.2. Đoạn chương trình sau sẽ in ra kết quả gì?

```
1. from random import *
2. A = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
3. C = set()
4. for i in range(5):
5.     x = choice(A)
6.     C.add(x)
7.     A.remove(x)
8. print(C)
```

So sánh kết quả này với bài tập 14.1.

Chương trình này chính là lời giải của bài tập 14.1.

14.3. Đoạn chương trình sau cho kết quả gì?

```
>>> import random as r
>>> C = {0, 1, 2, 3, 4, 5}
>>> x = r.choice(C)
>>> print(x)
```

a) 0

b) 5

c) Một giá trị ngẫu nhiên trong khoảng 0, 1, 2, 3, 4, 5.

d) Chương trình báo lỗi.

Đáp án đúng: d.

14.4. Giả sử cho tập hợp C như sau:

C = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}.

Viết chương trình tạo ra tập hợp A bao gồm 5 phần tử lấy ngẫu nhiên từ tập hợp C.

Gợi ý: Trước tiên chuyển tập hợp C về dạng list và sau đó dùng lời giải của bài tập 14.1.

14.5. Cho tập hợp C bao gồm các phần tử chỉ là số hoặc xâu ký tự. Viết chương trình tách tập hợp C thành 2 tập con N và S:

- N là tập con của C bao gồm các phần tử là số (nguyên hoặc thập phân).

- S là tập con của C bao gồm các phần tử là xâu ký tự.

```
# Bài 14.5
C = {1.2, 6, "One", "1010", 14, 4.5, "Tree", 70, 0}
N = set()
S = set()
for x in C:
    if str(type(x)) == "<class 'str'>":
        S.add(x)
    else:
        N.add(x)
print(C)
print(S)
print(N)
```

14.6. Giải thích ý nghĩa của chương trình sau:

```
C = set()
n = int(input("Nhập số tự nhiên n: "))
def isPrime(p):
    if p == 1:
        return False
    else:
        for k in range(2, p):
            if p % k == 0:
                return False
        return True
for k in range(1, n):
    if isPrime(k):
        C.add(k)
print(C)
```

Trả lời: Chương trình trên tạo tập hợp C bao gồm các số nguyên tố < n với n được nhập từ bàn phím.

14.7. Dựa trên kết quả của bài 6, viết chương trình thực hiện công việc sau:

- Nhập từ bàn phím số tự nhiên n.

- In ra màn hình dãy n số nguyên tố đầu tiên.

Chương trình của bài tập 14.6 là gợi ý cho lời giải bài tập này. Lời giải cần sửa một chút từ bài tập trên.

```
# Bài 14.7
C = set()
n = int(input("Nhập số tự nhiên n: "))
def isPrime(p):
    if p == 1:
        return False
    else:
        for k in range(2, p):
            if p % k == 0:
                return False
        return True
k = 1
count = 0
while count < n:
    if isPrime(k):
        C.add(k)
```

```
17.             count = count + 1
18.         k = k + 1
19. print(C)
```

14.8. Viết chương trình sinh ngẫu nhiên 2 tập hợp số tự nhiên A, B. Sau đó liệt kê ra màn hình các phần tử chung của A, B.

```
1.  # 14.8
2.  # Sinh ngẫu nhiên 2 tập hợp A, B. Tính giao của A, B.
3.  from random import randint
4.  from random import randint
5.  N = 30
6.  A = set()
7.  m = randint(20,40)
8.  for i in range(m):
9.      A.add(randint(1,N))
10. B = set()
11. m = randint(20,40)
12. for i in range(m):
13.     B.add(randint(1,N))
14. C = A & B
15. print(A)
16. print(B)
17. print("Giao 2 tập hợp:",C)
```

14.9. Viết chương trình sinh ngẫu nhiên 2 tập hợp số tự nhiên A, B. Sau đó liệt kê ra màn hình các phần tử thuộc A nhưng không thuộc B.

```
1.  # 14.9
2.  # Sinh ngẫu nhiên 2 tập hợp A, B. Tính hiệu của A, B.
3.  from random import randint
4.  N = 30
5.  A = set()
6.  m = randint(20,40)
7.  for i in range(m):
8.      A.add(randint(1,N))
9.  B = set()
10. m = randint(20,40)
11. for i in range(m):
12.     B.add(randint(1,N))
13. C = A - B
14. print(A)
15. print(B)
16. print("Hiệu 2 tập hợp:",C)
```

14.10. Viết chương trình sinh ngẫu nhiên 2 tập hợp số tự nhiên A, B. Sau đó liệt kê ra màn hình các phần tử hoặc thuộc A nhưng không thuộc B hoặc thuộc B nhưng không thuộc A.

```
1.  # 14.10
2.  # Sinh ngẫu nhiên 2 tập hợp A, B. Tính hiệu đối xứng của A, B.
3.  from random import randint
4.  N = 30
5.  A = set()
6.  m = randint(20,40)
7.  for i in range(m):
8.      A.add(randint(1,N))
```

```
9.  B = set()
10. m = randint(20,40)
11. for i in range(m):
12.     B.add(randint(1,N))
13. C = A ^ B
14. print(A)
15. print(B)
16. print("Hiệu đối xứng 2 tập hợp:",C)
```

14.11. Cho đoạn chương trình sau:

```
>>> A = {0, 2, 3, 6, 9, 10, 11}
>>> B = {0, 1, 4, 6, 7, 8, 9, 12}
>>> C = A&B
```

Tập hợp C bằng bao nhiêu?

a) {0, 1, 3, 6, 8, 9}

b) {0, 6, 9}

c) {0, 1, 6, 9, 10}

d) {0}

Đáp án đúng: b.

14.12. Cho đoạn chương trình sau:

```
>>> A = {0, 2, 3, 6, 9, 10, 11}
>>> B = {0, 1, 4, 6, 7, 8, 9, 12}
>>> C = A - B
```

Tập hợp C bằng bao nhiêu?

a) {0, 1, 3, 6, 8, 9, 10}

b) {0, 10, 6, 9}

c) {10, 11, 2, 3}

d) {0, 1, 2, 3}

Đáp án đúng: c.

14.13. Cho đoạn chương trình sau:

```
>>> A = {0, 2, 3, 6, 9, 10, 11}
>>> B = {0, 1, 4, 6, 7, 8, 9, 12}
>>> C = A^B
```

Tập hợp C bằng bao nhiêu?

a) {1, 2, 3, 4, 7, 8, 10, 11, 12}

b) {0, 1, 2, 3, 4, 6, 7, 8, 9}

c) {1, 2, 3, 4, 10, 11, 12, 3}

d) {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}

Đáp án đúng: a.

14.14. Quan sát chương trình sau:

```
n = int(input("Nhập số n: "))
C = set()
def isPrime(p):
    if p == 1:
        return False
    else:
        for k in range(2, p):
            if p % k == 0:
                return False
        return True
k = 2
while k <= n:
    if n%k == 0:
        C.add(k)
        while n%k == 0:
            n = n // k
    else:
        k = k + 1
print(C)
```

Kết quả tập hợp C chứa các giá trị nào?

Đáp án: C là tập hợp các số nguyên tố là ước số của n với n nhập từ bàn phím.

14.15. Viết chương trình sinh các tập hợp A, B, C như sau:

- A là tập hợp 100 số tự nhiên ngẫu nhiên trong khoảng [1, 1000].

- B là tập hợp 20 số tự nhiên ngẫu nhiên lấy từ tập hợp A.

- C là tập hợp 10 số tự nhiên ngẫu nhiên lấy từ tập hợp B.

```
# 14.15
# Sinh ngẫu nhiên tập hợp A. Từ A tạo B, C.
from random import choice
A = set()
B = set()
C = set()
for i in range(100):
    A.add(randint(1,1000))
A1 = A.copy()
for i in range(20):
    b = choice(list(A1))
    B.add(b)
    A1.remove(b)
A2 = A.copy()
for i in range(10):
    b = choice(list(A2))
    C.add(b)
    A2.remove(b)
print(A)
print(B)
print(C)
```

14.16. Viết chương trình sinh các tập hợp A, B có ý nghĩa như sau:

- A là tập hợp 10 số tự nhiên ngẫu nhiên lấy từ khoảng [2, 1000].

- B là tập hợp tất cả các số là ước số nguyên tố của các số lấy từ A. Điều đó có nghĩa là số p thuộc B khi và chỉ khi: p là số nguyên tố và p là ước số của một số nằm trong A. Ở đây hiểu ước số của một số có thể là 1 và chính nó.

```
# 14.16
from random import randint
A = set()
for i in range(10):
    A.add(randint(2,1000))
def isDivide(k):
    for n in A:
        if n%k == 0:
            return True
    return False
def nguyento(n):
    if n == 1:
        return True
    else:
        k = 2
        while k < n and n%k != 0:
            k = k + 1
        return k == n
B = set()
for k in range(2,1000):
    if nguyento(k) and isDivide(k):
        B.add(k)
print(A)
print(B)
```

Ví dụ một kết quả của chương trình trên:

```
{545, 227, 421, 904, 715, 46, 625, 337, 275, 511}
{2, 227, 5, 421, 7, 73, 11, 13, 109, 113, 337, 23}
```

14.17. Cho trước tập hợp C bao gồm cả số nguyên, số thực và xâu ký tự. Viết chương trình đếm số phần tử là nguyên, thập phân và xâu ký tự của C.

```
# 14.17
A = {1,2,1.5, 9.7, "One","Two","C",12,13,14,3.14, 1.24,"0"}
sint = 0
sfloat = 0
sstr = 0
for k in A:
    if type(k) == type(1):
        sint += 1
    if type(k) == type(1.0):
        sfloat += 1
    if type(k) == type("0"):
        sstr += 1
print(A)
print("Tập hợp A chứa:")
print(sint,"số nguyên",sfloat,"số thực",sstr, "xâu")
```

14.18. Viết chương trình thực hiện công việc sau:

- Nhập một số tự nhiên n từ bàn phím.

- Tạo 2 tập hợp A, B: A bao gồm các số nguyên tố đồng thời là ước số của n; B bao gồm các số nguyên tố nhỏ hơn n và không là ước của n.

```
# 14.18
n = int(input("Nhập số tự nhiên n: "))
A = set()
B = set()
def nguyento(n):
    if n == 1:
        return False
    else:
        k = 2
        while k < n and n%k != 0:
            k = k + 1
        return k == n
for k in range(2,n):
    if nguyento(k):
        if n%k == 0:
            A.add(k)
        else:
            B.add(k)
print("Tập hợp A:",A)
print("Tập hợp B:",B)
```

14.19. Giả sử có 2 tập hợp A, B. Để bổ sung các phần tử thuộc B nhưng nằm ngoài A đưa vào A thì dùng lệnh nào dưới đây:

a) A = A + B

b) A.union(B)

c) A | B

d) A = A.union(B)

Đáp án đúng: d)

14.20. Giả sử chúng ta đã có 2 tập hợp A, B. A là tập hợp tên các bạn học sinh lớp 9A, là họ tên các bạn học sinh lớp 9B. Ví dụ:

A = {"Bùi Việt Hà", "Nguyễn Quang Vinh", "Trương Anh Tú", "Trần Thị Hoa"}.

B = {"Nguyễn Việt Hà", "Bùi Cẩm Tú", "Lê Quang Vinh", "Hà Tú Anh"}.

Viết một chương trình tính:

- Số các học sinh có tên trùng nhau với lớp bên cạnh của 2 lớp. Trong ví dụ trên số học sinh trùng tên là 3 của mỗi lớp.

- Số các học sinh có họ trùng nhau với lớp bên cạnh của 2 lớp. Trong ví dụ trên số học sinh trùng họ của mỗi lớp là 2.

```
# 14.20
A = {"Bùi Việt Hà", "Nguyễn Quang Vinh", "Trương Anh Tú", "Trần Thị Hoa"}
B = {"Nguyễn Việt Hà", "Bùi Cẩm Tú", "Lê Quang Vinh", "Hà Tú Anh"}.
def Tach_ho(hoten):
    hoten = hoten.strip()
    slist = hoten.split()
```

```
7.      return slist[0]
8.  def Tach_ten(hoten):
9.      hoten = hoten.strip()
10.     slist = hoten.split()
11.     return slist[len(slist)-1]
12. Ah = set()
13. At = set()
14. Bh = set()
15. Bt = set()
16. for hoten in A:
17.     Ah.add(Tach_ho(hoten))
18.     At.add(Tach_ten(hoten))
19. for hoten in B:
20.     Bh.add(Tach_ho(hoten))
21.     Bt.add(Tach_ten(hoten))
22. Ch = Ah&Bh
23. Ct = At&Bt
24. print("2 lớp có",len(Ct),"tên chung:",Ct)
25. print("2 lớp có",len(Ch),"họ chung:",Ch)
```

14.21. Sử dụng tính chất các hàm set() và list() có thể chuyển đổi dữ liệu từ kiểu list sang set và ngược lại, viết các hàm số sau:

- Hàm f(<list>) với đầu vào là một list, đầu ra cũng là một list nhưng đã bỏ đi các phần tử giống nhau của dãy gốc.

- Hàm f(<list1>, <list2>) với đầu vào là 2 list bất kỳ, đầu ra là một dãy bao gồm tất cả các phần tử có trong 2 dãy gốc trên sau khi đã loại đi các phần tử giống nhau.

Các hàm số này được định nghĩa đơn giản như sau:

a. Hàm xóa bỏ đi các phần tử trùng nhau của một list.

```
def f(A):
    S = set(A)
    return list(S)
```

b. Hàm kết nối 2 list sau khi xóa đi các phần tử trùng nhau.

```
def f(A,B):
    SA = set(A)
    SB = set(B)
    return list(SA|SB)
```

Các chương trình hoàn chỉnh mô tả các hàm trên như sau.

Câu a).

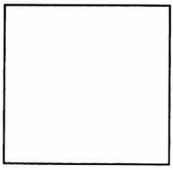
```
1.  # 14.21a
2.  A =
    [1,1,3,4,5,0,1,2,3,"One","Two","One","1","0","1","0",1,1,0,0]
3.  def f(A):
4.      S = set(A)
5.      return list(S)
6.  B = f(A)
7.  print(B)
```

Câu b).

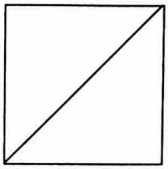
```
1. # 14.21b
2. A =
   [1,1,3,4,5,0,1,2,3,"One","Two","One","1","0","1","0",1,1,0,0]
3. B = [1,2,3,4,5,6,7,8,9,10,11,12,13]
4. def f(A,B):
5.     SA = set(A)
6.     SB = set(B)
7.     return list(SA|SB)
8. C = f(A,B)
9. print(C)
```

Chương 15: Đồ họa con rùa

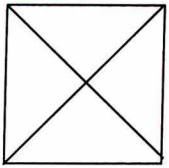
15.1. Viết chương trình cho rùa vẽ các hình sau:



(a)



(b)



(c)

Câu a).

```
1.  # Bài 15.1a
2.  from turtle import *
3.  shape("turtle")
4.  d = 100
5.  for i in range(4):
6.      forward(d)
7.      left(90)
8.  mainloop()
```

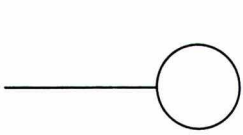
Câu b).

```
1.  # Bài 15.1b
2.  from turtle import *
3.  from math import sqrt
4.  shape("turtle")
5.  d = 100
6.  for i in range(4):
7.      forward(d)
8.      left(90)
9.  left(45)
10. forward(d*sqrt(2))
11. mainloop()
```

Câu c).

```
1.  # Bài 15.1c
2.  from turtle import *
3.  from math import sqrt
4.  shape("turtle")
5.  d = 100
6.  for i in range(4):
7.      forward(d)
8.      left(90)
9.  left(45)
10. forward(d*sqrt(2))
11. home()
12. forward(d)
13. left(135)
14. forward(d*sqrt(2))
15. mainloop()
```

15.2. Viết chương trình vẽ các hình sau:



(a)

(b)

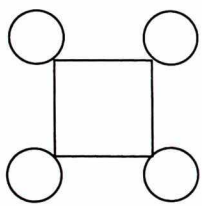
Câu a).

```
1.  # Bài 15.2a
2.  from turtle import *
3.  shape("turtle")
4.  d = 100
5.  R = 50
6.  forward(d)
7.  right(90)
8.  circle(R)
9.  mainloop()
```

Câu b).

```
1.  # Bài 15.2b
2.  from turtle import *
3.  shape("turtle")
4.  d = 100
5.  R = 50
6.  setheading(45)
7.  forward(d)
8.  right(90)
9.  circle(R)
10. mainloop()
```

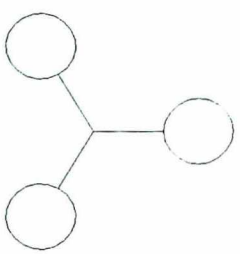
15.3. Viết chương trình vẽ hình sau:



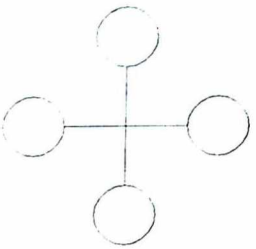
```
1.  # Bài 15.3
2.  from turtle import *
3.  shape("turtle")
4.  d = 100
5.  R = 30
6.  home()
7.  for i in range(4):
8.      forward(d)
9.      right(135)
10.     circle(R)
11.     left(135)
```

```
12.     left(90)
13. mainloop()
```

15.4. Viết chương trình vẽ các hình sau:



(a)



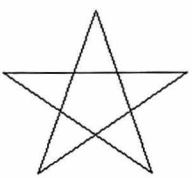
(b)

Xem lời giải 15.5.

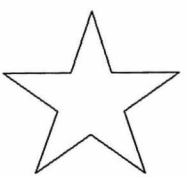
15.5. Tổng quát bài tập 4, vẽ hình bông hoa n cánh, mỗi cánh hoa là hình tròn.

```
1.  # Bài 15.5
2.  from turtle import *
3.  shape("turtle")
4.  n = 7
5.  d = 100
6.  R = 30
7.  home()
8.  for i in range(n):
9.      forward(d)
10.     right(90)
11.     circle(R)
12.     left(90)
13.     back(d)
14.     left(360/n)
15. mainloop()
```

15.6. Viết chương trình vẽ các hình ngôi sau sau đây.



(a)



(b)

Câu a).

```
1.  # Bài 15.6a. Vẽ ngôi sao 5 cánh
2.  # R - bán kính. Tâm tại gốc tọa độ (0,0)
3.  # d = độ dài cạnh lớn ngôi sao
```

```
# c = độ dài cạnh nhỏ ngôi sao
# Góc đỉnh = 18 độ

from turtle import *
from math import sin, cos, radians
shape("turtle")
R = 200
d = 2*R*cos(radians(18))
c = R*(1-sin(radians(18)))/cos(radians(18))
penup()
home()
setheading(90)
forward(R)
pendown()
for i in range(5):
    left(162)
    forward(d)
    right(18)
mainloop()
```

Câu b).

```
# Bài 15.6a. Vẽ ngôi sao 5 cánh, type 2
# R - bán kính. Tâm tại gốc tọa độ (0,0)
# d = độ dài cạnh lớn ngôi sao
# c = độ dài cạnh nhỏ ngôi sao
# Góc đỉnh = 18 độ

from turtle import *
from math import sin, cos, radians
shape("turtle")
R = 200
d = 2*R*cos(radians(18))
c = R*(1-sin(radians(18)))/cos(radians(18))
penup()
home()
setheading(90)
forward(R)
pendown()
for i in range(5):
    left(162)
    forward(c)
    right(72)
    forward(c)
    right(18)
mainloop()
```

15.7. Viết chương trình vẽ quốc kỳ Việt Nam, các thông số cụ thể như sau.

Một số thông số của lá cờ:

- Chiều rộng 200, chiều dài 300.

- Khoảng cách từ tâm đến 1 đỉnh của ngôi sao: 60.

Hay tổng quát:

- Chiều dài lá cờ dài gấp 3/2 chiều rộng.

- Khoảng cách từ tâm đến đỉnh ngôi sao bằng 1/5 chiều dài.

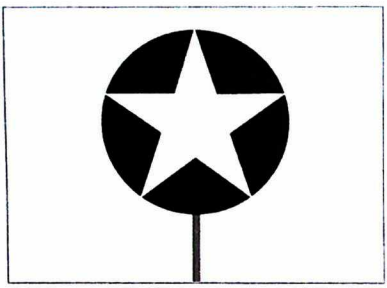
```
1.  # Bài 15.7. Vẽ lá cờ đỏ sao vàng 5 cánh
2.  # R - bán kính. Tâm tại gốc tọa độ (0,0)
3.  # d = độ dài cạnh lớn ngôi sao
4.  # c = độ dài cạnh nhỏ ngôi sa0
5.  # Góc đỉnh = 18 độ
6.
7.  from turtle import *
8.  from math import sin, cos, radians
9.  shape("turtle")
10. w = 300 # chiều rộng lá cờ
11. h = 200 # chiều cao lá cờ
12. R = 60  # bán kính ngôi sao
13. d = 2*R*cos(radians(18))
14. c = R*(1-sin(radians(18)))/cos(radians(18))
15. penup()
16. hideturtle()
17. # Vẽ lá cờ
18. goto(-w/2,-h/2)
19. color("red")
20. pendown()
21. begin_fill()
22. for i in range(2):
23.     forward(w)
24.     left(90)
25.     forward(h)
26.     left(90)
27. end_fill()
28. penup()
29. # vẽ ngôi sao
30. home()
31. setheading(90)
32. forward(R)
33. color("yellow")
34. pendown()
35. begin_fill()
```

```
for i in range(5):
    left(162)
    forward(c)
    right(72)
    forward(c)
    right(18)
end_fill()

mainloop()
```

15.8. Viết chương trình vẽ đèn ông sao năm cánh như hình sau.

Khoảng cách từ tâm đến đỉnh ngôi sao là 120 point.



Yêu cầu bắt buộc:

- Hình tròn màu đỏ.
- Ngôi sao màu vàng.
- Cán đèn ông sao màu xanh lá cây.

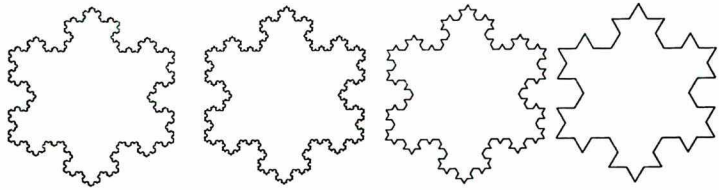
Chương trình cần viết đủ tổng quát để có thể thay đổi vị trí tâm và bán kính hình tròn.

```
# Bài 15.8. Vẽ đèn ông sao
# R - bán kính. Tâm tại gốc tọa độ (0,0)
# d = độ dài cạnh lớn ngôi sao
# c = độ dài cạnh nhỏ ngôi sao
# p = độ dài cán đèn ông sao
# Góc đỉnh = 18 độ

from turtle import *
from math import sin, cos, radians
shape("turtle")
R = 100
p = 60
d = 2*R*cos(radians(18))
c = R*(1-sin(radians(18)))/cos(radians(18))
penup()
hideturtle()
pensize(1)
# Vẽ hình tròn
goto(0,-R)
color("red")
pendown()
begin_fill()
circle(R)
end_fill()
penup()
# vẽ ngôi sao
home()
setheading(90)
```

```
29. forward(R)
30. color("yellow")
31. pendown()
32. begin_fill()
33. for i in range(5):
34.     left(162)
35.     forward(c)
36.     right(72)
37.     forward(c)
38.     right(18)
39. end_fill()
40. penup()
41. # vẽ cán đèn ông sao
42. pensize(5)
43. color("green")
44. goto(0,-R-p)
45. setheading(90)
46. pendown()
47. forward(p)
48. penup()
49.
50. mainloop()
```

15.9. Thiết kế chương trình chỉ cần 1 thủ tục có thể vẽ được các đường cong Koch với độ phức tạp khác nhau như các hình dưới đây.



Yêu cầu viết 1 thủ tục có dạng Koch-Curve với tham số là **level** - số mức của đường cong này. Tương ứng hình trên là các mức level = 5, 4, 3, 2 của thủ tục này.

Lời giải.

Từ cấu trúc của đường con Koch chúng ta thấy tổng thể đường cong sẽ bao gồm 3 nhánh nối lại với nhau từ 3 đỉnh của một tam giác đều. Trên mỗi cạnh của tam giác đó chính là một đường cong fractal Koch. Thủ tục chính vẽ đường cong này là ***KochCurve(d, level)*** *với d là độ dài cạnh lớn, level là mức của đường cong. Nếu level = 0 thì đường cong chỉ là đoạn thẳng độ dài d. Khi level > 0, hàm* ***KochCurve()*** *sẽ gọi đệ quy theo các mức level thấp dần, cho đến khi level = 0 thì dừng lại.*

```
1.  # 15.9
2.  # Vẽ đường cong fractal KochCurve
3.  from turtle import *
4.
5.  d = 250     # độ dài cạnh tam giác ngoài
6.  level = 6   # mức độ chi tiết của fractal
7.
8.  def KochCurve(d,level):
9.      if level == 0:
```

```
def veKhung():
    reset()
    hideturtle()
    penup()
    goto(-WIDTH//2,-HEIGHT//2)
    setheading(0)
    pendown()
    for i in range(2):
        forward(WIDTH)
        left(90)
        forward(HEIGHT)
        left(90)
    penup()

def init():
    penup()
    shape("turtle")
    shapesize(1.5)
    showturtle()
    goto(0,0)
    setheading(0)

def gogoxy(x,y):
    alpha = towards(x,y)
    setheading(alpha)
    v = distance(x,y)/10
    global run
    if not run:
        run = True
        Running(v)

def Running(v):
    while True:
        forward(v)
        if xcor() > WIDTH//2:
            alpha = heading()
            if alpha > 0:
                setheading(180-alpha)
            else:
                setheading(alpha - 180)
        if xcor() < -WIDTH//2:
            alpha = heading()
            setheading(180-alpha)
        if ycor() > HEIGHT//2:
            alpha = heading()
            setheading(-alpha)
        if ycor() < -HEIGHT//2:
            alpha = heading()
            setheading(-alpha)
veKhung()
init()
onscreenclick(gogoxy)
mainloop()
```

Chương 16: Bắt lỗi và kiểm soát lỗi chương trình

16.1. Viết đoạn chương trình yêu cầu nhập một số nguyên từ bàn phím, trong đó sử dụng bẫy lỗi (Exception), nếu người dùng nhập số thực hoặc văn bản. Nếu nhập sai, chương trình sẽ thông báo liên tục "Bạn cần nhập số nguyên" cho đến khi nhập đúng thì thôi.

```
# Bài 16.1
while True:
    try:
        n = int(input("Nhập một số nguyên: "))
        break
    except ValueError:
        print("Nhập sai rồi! Bạn phải nhập một số nguyên.")

print("Số bạn đã nhập:",n)
```

16.2. Mô tả ý nghĩa đoạn chương trình sau:

```
i = 0
Sum = 0
try:
  while True:
    Sum = Sum + lst[i]
    i = i + 1
except IndexError:
  pass
print("Tổng dãy số là =", Sum)
```

Viết lại đoạn chương trình trên không dùng bẫy lỗi Exception.

Lời giải.

Đoạn chương trình trên dùng bẫy lỗi Exception để dừng chương trình nếu gặp lỗi chỉ số dãy (khi giá trị i vượt quá chỉ số của dãy lst).

Có thể sửa lại không dùng bẫy Exception như sau.

```
# Bài 16.2
lst = [1,2,3,4,5,6]
S = 0
for i in range(len(lst)):
    S = S + lst[i]
print("Tổng dãy số là =", S)
```

16.3. Mệnh đề nào dưới đây mô tả đúng ý nghĩa của bẫy lỗi Exception?

a) Dùng bẫy lỗi để tìm lỗi logic trong chương trình.

b) Dùng bẫy lỗi để tìm lỗi syntax trong chương trình.

c) Dùng bẫy lỗi để nhanh chóng tìm ra lỗi khi người dùng nhập dữ liệu.

d) Dùng bẫy lỗi để tránh cho chương trình bị dừng lại nếu gặp lỗi trong khi đang chạy.

Đáp án đúng: d.

16.4. Dùng một lệnh **try .... exception** có thể bẫy được bao nhiêu loại lỗi:

a) 1

b) 2

c) Không hạn chế.

d) Tùy vào từng trường hợp cụ thể.

Đáp án đúng: c.

16.5. Chương trình sau có thể gây lỗi logic như thế nào:

```
Sum = 0
f = open('data.dat','r')
for line in f:
  Sum = Sum + int(line)
f.close()
print(Sum)
```

Muốn bắt lỗi trong chương trình trên cần sửa như thế nào?

Lời giải.

Các lỗi có thể có của chương trình trên:

*- Tệp "data.dat" không tồn tại trên thư mục đã chọn. Lỗi này thuộc loại **FileNotFoundError**.*

*- Tệp đó tồn tại nhưng có dòng chứa số thập phân, chứa nhiều số hoặc chứa một xâu ký tự dài. Lỗi này thuộc loại **ValueError**.*

Chương trình sửa lại để bắt tất cả các lỗi trên sẽ như sau.

```
1.  # Bài 16.5
2.  Sum = 0
3.  fdata = "data.dat"
4.  try:
5.      f = open(fdata,'r')
6.      for line in f:
7.          Sum = Sum + int(line)
8.      f.close()
9.  except FileNotFoundError:
10.     print("Tệp",fdata,"không tồn tại.")
11. except ValueError:
12.     print("Dữ liệu trong tệp không hợp lệ.")
13. else:
14.     print(Sum)
```

16.6. Đoạn lệnh sau sẽ báo lỗi loại gì?

```
while True print(Hôm nay là ngày đẹp trời)
```

a) Lỗi Exception.

b) Lỗi Syntax.

c) Lỗi Logic.

d) Không báo lỗi.

Đáp án đúng: b.

16.7. Hãy sửa và thêm các lệnh bắt lỗi vào đoạn chương trình sau.

```
def Change(A, p):
  for i in range(p):
      A[i] = 0
```

*Chương trình trên sẽ báo lỗi **IndexError** nếu p > len(A).*

Có thể sửa lại như sau:

```
def Change(A, p):
    try:
        for i in range(p):
            A[i] = 0
    except IndexError:
        pass
```

Xét ví dụ chương trình sau:

```
1.  # Bài 16.7
2.  A = [0,1,2,3,4,5,6]
3.  p = 10
4.  def Change(A, p):
5.      try:
6.          for i in range(p):
7.              A[i] = 0
8.      except IndexError:
9.          pass
10. Change(A,p)
11. print(A)
```

16.8. Giả sử chúng ta đã định nghĩa một hàm f(x), sau đó thực hiện lời gọi hàm, ví dụ như các lệnh sau:

```
1. def f(x):
2.     <thân hàm số>
3.    return <giá trị hàm số>
4. ......
5. a = f(10)
```

Để kiểm soát lỗi tại dòng 5, chúng ta sẽ dùng mã lỗi Exception nào?

a) ValueError.

b) TypeError.

c) SyntaxError.

d) NameError.

Đáp án đúng: a; b.

16.9. Các lỗi Exception trong Python thuộc kiểu dữ liệu gì?

a) Tuple

b) List

c) Là các lớp con của lớp Exception

d) Không thuộc kiểu dữ liệu nào.

Đáp án đúng: c.

16.10. Các mã lỗi Exception do người dùng tự định nghĩa cần phải thỏa mãn điều kiện gì?

a) Cần có tên kết thúc bằng cụm từ Error.

b) Là lớp con của lớp Exception.

c) Là lớp con của lớp BaseException.

Đáp án đúng: b.

16.11. Khi dùng lệnh **raise** để tự tạo lỗi, có thể dùng mã lỗi thông thường hay không?

a) Có thể được.

b) Không thể, bắt buộc phải tự định nghĩa một mã lỗi riêng.

Đáp án đúng: a.

16.12. Hàm số GenList(n) có chức năng yêu cầu người dùng nhập và trả về danh sách (list) có n phần tử là số tự nhiên. Viết hàm số này với yêu cầu bắt lỗi Exception chính xác.

*Hàm **GenList** cần bắt được tất cả các lỗi sau:*

- Người dùng nhập không là số nguyên, ví dụ nhập một xâu, một số thực hoặc một dãy nhiều số. Lỗi này Python tự động sinh mã lỗi ValueError.

- Người dùng nhập số nguyên không là số tự nhiên, ví dụ nhập số nhỏ hơn hoặc bằng 0. Lỗi này Python không sinh lỗi nên chúng ta phải tự tạo lỗi ValueError. Một hàm riêng NhapTN(k) sẽ được thiết lập để tự tạo lỗi ValueError nếu người dùng nhập một số nguyên nhưng không là số tự nhiên.

```
1.  # Bài 16.12
2.  def nhapTN(k):
3.      m = int(input("Nhập số tự nhiên thứ " + str(k) + ": "))
4.      if m <= 0:
5.          raise ValueError
6.      return m
7.  def GenList(n):
8.      A = []
9.      for i in range(1,n+1):
10.         while True:
11.             try:
12.                 m = nhapTN(i)
13.                 break
14.             except ValueError:
15.                 print("Nhập không chính xác, hãy nhập lại.")
16.         A.append(m)
17.     return A
18. A = GenList(6)
19. print("Dãy số tự nhiên đã nhập là:")
20. print(A)
```

Tài liệu tham khảo

[1]. Tài liệu gốc của Python. https://www.python.org/doc/

[2]. Fundamental of Python Programming. Richard L. Halterman. 2019.

[3]. Invent Your Own Computer Games with Python. 2nd Edition. Al Sweigart.

[4]. Python® Notes for Professionals. https://goalkicker.com/PythonBook/

[5]. LOGO Programming Book.

[6]. Introduction to Algorithms. Third Edition. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein.

[7]. Discrete Mathematics and Its Applications. 7th Edition. Kenneth H. Rosen.

[8]. Thiết kế phần mềm và trò chơi giáo dục. Bùi Việt Hà. NXB Đại học Quốc gia Hà Nội. 2018.

[9]. Thiết kế trò chơi Lines. Bùi Việt Hà.

[10]. STEM Computing. Bùi Việt Hà.

[11]. Các bản tin CS4S 2017, 2018, 2019.

[12]. Introduction to The Design and Analysis of Algorithm. 3rd Edition. Anany Levitin.

[13]. Computing in the national curriculum. A guide for primary teachers.

[14]. Computing in the national curriculum. A guide for secondary teachers.

[15]. Learn to Program with Scratch. Majed Marji.

[16]. Scratch 2.0 Game Development. Sergio van Pul, Jessica Chiang.

[17]. Computer Science Concepts in Scratch. Michal Armoni and Moti BenAri, 2013. http://stwww.weizmann.ac.il/g-cs/scratch/scratch-14-textbook-1-0-one-side.pdf

[18]. Program or be Programmed: Ten Commands for a Digital Age. Rushkoff, D. OR Books, 2009.

[19]. CS unplugged. Giảng dạy khoa học máy tính không cần máy tính. http://www.csunplugged.org

[20]. Chương trình, chuẩn kiến thức môn Tin học. NXB Giáo dục, Hà Nội, 2000.

[21]. Chương trình giáo dục phổ thông tổng thể trong chương trình giáo dục phổ thông mới. Tài liệu chính thức Bộ Giáo dục và Đào tạo, 2017.

[22]. Computing in the national curriculum A guide for primary teachers. Tài liệu dành cho GV cấp Tiểu học Anh quốc dạy môn Tin học.
http://www.computingatschool.org.uk/data/uploads/CASPrimaryComputing.pdf

[23]. Shut Down or Restart.
https://royalsociety.org/topics-policy/projects/computing-in-schools/report/
https://royalsociety.org/~/media/education/computing-in-schools/2012-01-12-computing-in-schools.pdf

[24]. Computing at School, The Raspberry Pi Education Manual (CAS, 2012). http://pi.cs.man.ac.uk/download/Raspberry_Pi_Education_Manual.pdf

[25]. Scratch Programming For Teens. Jerry Lee Ford, Jr.

[26]. Super Scratch Programming Adventure! The LEAD Project

[27]. Computational thinking. A guide for teachers. Andrew Csizmadia, Simon Humphreys, National Coordinator, Computing At School.

[28]. Computer Science: A curriculum for schools.

http://www.computingatschool.org.uk/

[29]. K12-Computer Science Framework.

https://k12cs.org/wp-content/uploads/2016/09/K%E2%80%9312-Computer-Science-Framework.pdf.

[30]. Lịch sử phát triển môn Tin học ở Việt Nam. CS4S số 3 (2017).

[31]. Đề cương khung chương trình môn Tin học của Anh quốc (bản mới). CS4S số 1 (2017).

[32]. Hiểu và đi trong cách mạng công nghiệp lần thứ tư. Hồ Tú Bảo. 2017.

[33]. Tự học lập trình Scratch. Bùi Việt Hà. NXB Đại học Quốc gia Hà Nội. 2018.

[34]. Thiết kế phần mềm mô phỏng chơi Cờ Vua trên Scratch. Bùi Việt Hà.

[35]. O-level Computing for teaching and learning. Curriculum Planning and Development Division.

[36]. Kodu for Kids. QUE Publication.

[37]. Oxford International Primary Computing. Volume 1-6. Oxford University Press.

[38]. Góp ý cho đề cương môn Tin học trong Chương trình Giáo dục Phổ thông mới 2018. CS4S số 3 (2018).

[39]. Chương trình khung (chính thức) môn Tin học trong Chương trình Giáo dục Phổ thông mới. Bộ Giáo dục và Đào tạo. 7-2018.

[40]. Nhận xét, phân tích, góp ý cho Chương trình môn Tin học trong CTGDPT mới. Bùi Việt Hà. CS4S số 7 - 2018.

[41]. Nhận xét, góp ý cho Chương trình môn Tin học trong CTGDPT mới. CS4S số 9 - 2018.

[42]. Chương trình Giáo dục Phổ thông mới môn Tin học, phiên bản tháng 9-2018. Bộ Giáo dục và Đào tạo.

[43]. A smarter way to learn Python. Mark Myers. San Bernardino, CA.

[44]. Python Algorithms. Mastering Basic Algorithms in the Python Language. Magnus Lie Hetland. Apress. 2010.

[45]. Data Structure and Algorithms with Python. Kent D.Lee, Steve Hubbard. Springer Publisher. 2015.

NHÀ XUẤT BẢN
ĐẠI HỌC QUỐC GIA HÀ NỘI
16 Hàng Chuối - Hai Bà Trưng - Hà Nội

Giám đốc – Tổng Biên tập: (024)39715011
Hành chính: (024)39714899; Fax: (024)39724736
Quản lý xuất bản: (024) 39728806
Biên tập: (024) 39714896

Chịu trách nhiệm xuất bản
Giám đốc - Tổng biên tập: TS. Phạm Thị Trâm

Biên tập: NGUYỄN THỊ THÙY
Chế bản: BÙI VIỆT HÀ
Vẽ bìa: NGUYỄN NGỌC ANH

Đối tác: Công Ty TNHH Tin học và Nhà Trường
Địa chỉ: Nhà 6, ngõ 518, phố Đội Cấn, phường Cống Vị, quận Ba Đình, Hà Nội

LỜI GIẢI BÀI TẬP PYTHON CƠ BẢN

Mã số: 1L - 60ĐH2021
In 1.000 bản, khổ 19x27 tại Công ty Cổ phần in và Thương mại Ngọc Hưng
Địa chỉ: Số 296 đường Phúc Diễn, tổ dân phố số 1, P. Xuân Phương, Q. Nam Từ Liêm, Hà Nội
Số xuất bản: 129-2021/CXBIPH/08-04/ĐHQGHN, ngày 13/01/2021.
Quyết định xuất bản số: 25LK-TN/QĐ - NXB ĐHQGHN ngày 14/01/2021.
In xong và nộp lưu chiểu năm 2021.

Tủ sách

CS4S

Thời gian gần đây, cùng với phong trào STEM phát triển mạnh, các ngôn ngữ lập trình trực quan lần lượt ra đời như Scratch đã tạo ra một cuộc cách mạng mới, bùng nổ của giáo dục Tin học trong nhà trường. Lần đầu tiên trong lịch sử, định hướng Khoa học máy tính của môn Tin học được đưa xuống cho học sinh bậc Tiểu học. Cuộc cách mạng này sẽ có ý nghĩa to lớn và lâu dài mà chúng ta có lẽ chưa thể đánh giá hết ngày hôm nay. Rất may mắn là Python lại chính là ngôn ngữ lập trình thích hợp dùng trong nhà trường, cho lứa tuổi cấp THCS, THPT, phù hợp với cả 2 định hướng phát triển chính của môn Tin học hiện nay là STEM và ALGORITHM.

PYTHON